

An Epistemic Strategy Logic*

Xiaowei Huang Ron van der Meyden

The University of New South Wales

Abstract

The paper presents an extension of temporal epistemic logic with operators that quantify over agent strategies. Unlike previous work on alternating temporal epistemic logic, the semantics works with systems whose states explicitly encode the strategy being used by each of the agents. This provides a natural way to express what agents would know were they to be aware of the strategies being used by other agents. A number of examples that rely upon the ability to express an agent's knowledge about the strategies being used by other agents are presented to motivate the framework, including reasoning about game theoretic equilibria, knowledge-based programs, and information theoretic computer security policies. Relationships to several variants of alternating temporal epistemic logic are discussed. The computational complexity of model checking the logic and several of its fragments are also characterized.

1 Introduction

In distributed and multi-agent systems, agents typically have a choice of actions to perform, and have individual and possibly conflicting goals. This leads agents to act strategically, attempting to select their actions over time so as to guarantee achievement of their goals even in the face of other agents' adversarial behaviour. The choice of actions generally needs to be done on the basis of *imperfect* information concerning the state of the system.

These concerns have motivated the development of a variety of modal logics that aim to capture aspects of such settings. One of the earliest, dating from the 1980's, was multi-agent epistemic logic [21, 41], which introduces modal operators that deal with imperfect information by providing a way to state what agents *know*. Combining such constructs with temporal logic [43] constructs gives temporal-epistemic logics, which support reasoning about how agents' knowledge changes over time. Temporal-epistemic logic is an area about which a significant amount is now understood [17].

Logics dealing with reasoning about strategies, which started to be developed in the same period [40], had a slower initial start, but have in recent years become the focus

*This paper combines results from *An epistemic strategy logic*, X. Huang and R. van der Meyden, 2nd International Workshop on Strategic Reasoning, April 2014, Grenoble, France, and *A temporal logic of strategic knowledge*, X. Huang and R. van der Meyden, Int. Conf. on Principles of Knowledge Representation and Reasoning, Jul 2014, Vienna. It extends these works by including full proofs for all results.

of intense study [42, 27, 1]. *Alternating temporal logic* (ATL) [1], which generalizes branching time temporal logic to encompass reasoning about the temporal effects of strategic choices by one group of agents against all possible responses by their adversaries, has become a popular basis for work in this area.

One of the ways in which recent work has extended ATL is to add epistemic operators, yielding an *alternating temporal epistemic logic*. Many subtle issues arise concerning what agents know in settings where multiple agents act strategically. In the process of understanding these issues, there has been a proliferation of ATL extensions. In particular, for dealing with epistemic reasoning in strategic settings, there are multiple proposals, with different semantic and syntactic bases, for how to capture reasoning about the availability to groups of agents of strategies for achieving particular goals [35, 47, 39, 30, 31].

Some of the modal operators introduced in this literature are complex, interweaving ideas about the knowledge of a group of agents, the strategies available to them, the effect of playing these strategies against strategies available to agents not in the group, and the knowledge that other groups of agents may have about these effects. One of our objectives in the present paper is to identify a minimal set of primitives that can be composed to represent the more complex notions involving reasoning about strategies and knowledge that are useful for applications.

Our approach to this begins by treating agent strategies as first class citizens in the semantics, represented as components of the global state of the system at any moment of time. This is in contrast to many works in the area, where strategies are used to generate runs of a system, but where the runs themselves contain no explicit information about the specific strategies used by the players to produce them. Semantics that explicitly encode strategies in runs have been used previously in the literature [23]; what is novel in our approach is to develop a logic that enables explicit reference to these strategies.

In particular, by encoding strategies as components of the state, we may refer to the strategy in use by a specific agent. We introduce a syntactic notation $\sigma(i)$, that refers to the strategy of agent i , and allow this construct to be used in the same contexts where the agent name i can be used. We show that even this simple extension of temporal epistemic logic already gives a logical approach with broad applicability. In particular, it can express many of the subtly different notions that have been the subject of proposals for alternating temporal epistemic logics. We demonstrate this by results that show how such logics can be translated into our setting. We also present a number of other examples including reasoning about possible implementations of knowledge-based programs, game theoretic solution concepts, and issues of concern in computer security.

In some cases, however, some richer expressiveness is required to capture notions that arise in our applications in settings where agents reason about their common knowledge. We address this by adding to the logic constructs that can be used to express quantification over strategies. This leads to a logic which, like *strategy logic* [11, 38], supports explicit naming and quantification over strategies. However we achieve this in a slightly more general way: we first generalize temporal epistemic logic to include operators for quantification over global states and reference to their components, and then apply this generalization to a system that includes strategies encoded in the global

states and references these using the “strategic” agents of [29]. The resulting framework is able to deal with our applications when these require agents to reasoning about common knowledge.

While adding quantification adds desirable expressiveness to the logic, we show that it does come at a cost of complexity. We study this issue from the perspective of model checking. We consider several dimensions: does the logic have quantifiers, and what is the temporal basis for the logic: branching time (CTL) or linear time (LTL). The richest of logics in our spectrum turn out to have EXSPACE-complete model checking problems, but we identify a number of special cases where model checking is in PSPACE.

The structure of the paper is as follows. In Section 2, we first develop an extension of temporal epistemic logic that adds the ability to quantify over global states and refer to global state components. We then present a semantic model (also standard) for the environments in which agents choose their actions. Building on this model, we show how to construct a model for temporal epistemic logic called *strategy space* that builds in information about the strategy being used by each of the agents. We then define a spectrum of logics defined over the resulting semantics. These logics are obtained as fragments of the extended temporal epistemic logic, interpreted in in strategy space. One point on the spectrum adds new “strategy” names that refer to the strategy being played by an agent into an otherwise standard temporal epistemic logic. The richest point on the spectrum has the capacity to quantify over agent strategies. Next, in Section 3, we provide results on the complexity of the model checking problem for the various fragment of the logic, identifying fragments with lower complexity than the general problem. Section 4 deals with applications of the resulting logics. We show successively that they can express reasoning about implementations of knowledge-based programs, many notions that have been proposed in the area of alternating temporal epistemic logic, game theoretic solution concepts, and problems from computer security. We take care in the applications to use the fragments with lower model checking complexity whenever possible. In Section 5, we conclude with a discussion of related literature.

2 An extended temporal epistemic logic

We begin by defining the syntax and semantics of an extension of temporal epistemic logic that adds the ability to quantify over global states and refer to global state components. This syntax and semantics will be instantiated in what follows by taking some of the global state components to be the strategies being used by agents.

The usual *interpreted systems* semantics for temporal epistemic logic [17] deals with runs, in which each moment of time is associated with a global state that is comprised of a local state for each agent in the system. To quantify over global states, we extend temporal epistemic logic with a set of variables Var , an operator $\exists x$, and a construct $e_i(x)$, where x is a variable. The formula $\exists x.\phi$ says, intuitively, that there exists in the system a global state x such that ϕ (a formula that may contain uses of the variable x) holds at the current point. The formula $e_i(x)$ says that agent i has the same local state at the current point and at the global state x .

Let $Prop$ be a set of atomic propositions and let Ags be a set of agents. Formally, the language $ETLK(Ags, Prop, Var)$ has syntax given by the grammar:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid A\phi \mid \bigcirc\phi \mid \phi_1 U \phi_2 \mid \exists x.\phi \mid e_i(x) \mid D_G\phi \mid C_G\phi$$

where $p \in Prop$, $x \in Var$, $i \in Ags$, and $G \subseteq Ags$. The construct $D_G\phi$ expresses that agents in G have distributed knowledge of ϕ , i.e., could deduce ϕ if they pooled their information, and $C_G\phi$ says that ϕ is common knowledge to group G . The temporal formulas $\bigcirc\phi$, $\phi_1 U \phi_2$, $A\phi$ have the same intuitive meanings as in the temporal logic CTL^* [15], i.e., $\bigcirc\phi$ says that ϕ holds at the next moment of time, $\phi_1 U \phi_2$ says that ϕ_1 holds until ϕ_2 does, and $A\phi$ says that ϕ holds in all possible evolutions from the present situation.

Other operators can be defined in the usual way, e.g., $\phi_1 \wedge \phi_2 = \neg(\neg\phi_1 \vee \neg\phi_2)$, $\Diamond\phi = (true U \phi)$, $\Box\phi = \neg\Diamond\neg\phi$, etc. The universal form $\forall x.\phi = \neg\exists x.\neg\phi$ expresses that ϕ holds for all global states x . For an agent $i \in Ags$, we write $K_i\phi$ for $D_{\{i\}}\phi$; this expresses that agent i knows the fact ϕ . The notion of everyone in group G knowing ϕ can then be expressed as $E_G\phi = \bigwedge_{i \in G} K_i\phi$. We write $e_G(x)$ for $\bigwedge_{i \in G} e_i(x)$. This says that at the current point, the agents in G have the same local state as they do at the global state named by variable x .

We will be interested in a fragment of the logic that restricts the occurrence of the temporal operators to some simple patterns, in the style of the branching time temporal logic CTL [13]. We write $ECTL(Ags, Prop, Var)$ for the fragment of $ETLK(Ags, Prop, Var)$ in which the temporal operators occur only in the particular forms $A \bigcirc \phi$, $\neg A \neg \bigcirc \phi$, $A\phi_1 U \phi_2$, and $\neg A \neg \phi_1 U \phi_2$.

The semantics of $ETLK(Ags, Prop, Var)$ builds straightforwardly on the following definitions used in the standard semantics for temporal epistemic logic [17]. Consider a system for a set of agents Ags . A *global state* is an element of the set $\mathcal{G} = L_e \times \prod_{i \in Ags} L_i$, where L_e is a set of states of the environment and each L_i is a set of *local states* for agent i . A *run* is a mapping $r : \mathbb{N} \rightarrow \mathcal{G}$ giving a global state at each moment of time. A *point* is a pair (r, m) consisting of a run r and a time m . An *interpreted system* is a pair $\mathcal{I} = (\mathcal{R}, \pi)$, where \mathcal{R} is a set of runs and π is an *interpretation*, mapping each point (r, m) with $r \in \mathcal{R}$ to a subset of $Prop$. For $n \leq m$, write $r[n \dots m]$ for the sequence $r(n)r(n+1) \dots r(m)$. Elements of $\mathcal{R} \times \mathbb{N}$ are called the *points* of \mathcal{I} . For each agent $i \in Ags \cup \{e\}$, we write $r_i(m)$ for the component of $r(m)$ in L_i , and then define an equivalence relation on points by $(r, m) \sim_i (r', m')$ if $r_i(m) = r'_i(m')$. We also define $\sim_G^D \equiv \bigcap_{i \in G} \sim_i$, and $\sim_G^E \equiv \bigcup_{i \in G} \sim_i$, and $\sim_G^C \equiv (\bigcup_{i \in G} \sim_i)^*$ for $G \subseteq Ags$. We take \sim_\emptyset^D to be the universal relation on points, and (for the sake of preserving monotonicity of these relations in these degenerate cases) take \sim_\emptyset^E and \sim_\emptyset^C to be the identity relation.

To extend this semantic basis for temporal epistemic logic to a semantics for $ETLK(Ags, Prop, Var)$, we just need to add a construct that interprets variables as global states. A *context* for an interpreted system \mathcal{I} is a mapping Γ from Var to global states occurring in \mathcal{I} . We write $\Gamma[g/x]$ for the context Γ' with $\Gamma'(x) = g$ and $\Gamma'(y) = \Gamma(y)$ for all variables $y \neq x$. The semantics of the language $ETLK$ is given by a relation $\Gamma, \mathcal{I}, (r, m) \models \phi$, representing that formula ϕ holds at point (r, m) of the interpreted system \mathcal{I} , relative to context Γ . This is defined inductively on the structure of the formula ϕ , as follows:

- $\Gamma, \mathcal{I}, (r, m) \models p$ if $p \in \pi(r, m)$;
- $\Gamma, \mathcal{I}, (r, m) \models \neg\phi$ if not $\Gamma, \mathcal{I}, (r, m) \models \phi$;
- $\Gamma, \mathcal{I}, (r, m) \models \phi \wedge \psi$ if $\Gamma, \mathcal{I}, (r, m) \models \phi$ and $\Gamma, \mathcal{I}, (r, m) \models \psi$;
- $\Gamma, \mathcal{I}, (r, m) \models A\phi$ if $\Gamma, \mathcal{I}, (r', m) \models \phi$ for all $r' \in \mathcal{R}$ with $r[0 \dots m] = r'[0 \dots m]$;
- $\Gamma, \mathcal{I}, (r, m) \models \bigcirc\phi$ if $\Gamma, \mathcal{I}, (r, m+1) \models \phi$;
- $\Gamma, \mathcal{I}, (r, m) \models \phi U \psi$ if there exists $m' \geq m$ such that $\Gamma, \mathcal{I}, (r, m') \models \psi$ and $\Gamma, \mathcal{I}, (r, k) \models \phi$ for all k with $m \leq k < m'$;
- $\Gamma, \mathcal{I}, (r, m) \models \exists x.\phi$ if $\Gamma[r'(m')/x], \mathcal{I}, (r, m) \models \phi$ for some point (r', m') of \mathcal{I} ;
- $\Gamma, \mathcal{I}, (r, m) \models e_i(x)$ if $r_i(m) = \Gamma(x)_i$;
- $\Gamma, \mathcal{I}, (r, m) \models D_G\phi$ if $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all (r', m') such that $(r', m') \sim_G^D (r, m)$;
- $\Gamma, \mathcal{I}, (r, m) \models C_G\phi$ if $\Gamma, \mathcal{I}, (r', m') \models \phi$ for all (r', m') such that $(r', m') \sim_G^C (r, m)$.

The definition is standard, except for the constructs $\exists x.\phi$ and $e_i(x)$. The clause for the former says that $\exists x.\phi$ holds at a point (r, m) if there exists a global state $g = r'(m')$ such that ϕ holds at the point (r, m) , provided, we interpret x as referring to g . Note that it is required that g is attained at some point (r', m') , so actually occurs in the system \mathcal{I} . The clause for $e_i(x)$ says that this holds at a point (r, m) if the local state of agent i , i.e., $r_i(m)$, is the same as the local state $\Gamma(x)_i$ of agent i at the global state $\Gamma(x)$ that interprets the variable x according to Γ .

We remark that these novel constructs introduce some redundancy, in that the set of epistemic operators D_G could be reduced to the “universal” operator D_\emptyset , since $D_G\phi \equiv \exists x.(e_G(x) \wedge D_\emptyset(e_G(x) \Rightarrow \phi))$. Evidently, given the syntactic complexity of this formulation, D_G remains a useful notation.

2.1 Strategic Environments

In order to semantically represent settings in which agents operate by strategically choosing their actions in the context of an environment, we introduce a type of transition system that models the available actions and their effects on the state. We generate from this transition system an instance of the interpreted systems semantics defined in the previous section. One of the innovations in this construction is to introduce new names, that refer to global state components that represent the strategies being used by the agents.

An *environment* for agents Ags is a tuple $E = \langle S, I, \{Acts_i\}_{i \in Ags}, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$, where

1. S is a set of states,
2. I is a subset of S , representing the initial states,
3. for each $i \in Ags$, component $Acts_i$ is a nonempty set of actions that may be performed by agent i ; we define $Acts = \prod_{i \in Ags} Acts_i$ to be the set of joint actions,

4. component $\rightarrow \subseteq S \times Acts \times S$ is a transition relation, labelled by joint actions,
5. for each $i \in Ags$, component $O_i : S \rightarrow L_i$ is an observation function, and
6. $\pi : S \rightarrow \mathcal{P}(Prop)$ is a propositional assignment.

An environment is said to be finite if all its components, i.e., $S, Ags, Acts_i, L_i$ and $Prop$ are finite. Intuitively, a joint action $a \in Acts$ represents a choice of action $a_i \in Acts_i$ for each agent $i \in Ags$, performed simultaneously, and the transition relation resolves this into an effect on the state. We assume that \rightarrow is serial in the sense that for all $s \in S$ and $a \in Acts$ there exists $t \in S$ such that $(s, a, t) \in \rightarrow$.

For purposes of results concerning the complexity of model checking, we need a measure of the size of a finite environment. Conventionally, the size of a model is taken to be the length of a string that lists its components, and typically, this is polynomial in the number of states of the model. However, we note that in the case of environments, the set of labels $Acts$ of the transition relation is an n -fold cartesian product, where $n = Ags$, so this may already be of exponential size in the size of the other components. To address this, we allow that the transition relation \rightarrow is presented in some notation with the property that, given states s, t and a joint action a , determining whether $s \xrightarrow{a} t$ is in PTIME. One example of a presentation format with this property is the class of *turn-based environments*, where at each state s , there exists an agent i such that if $s \xrightarrow{a} t$ for a joint action a , then for all joint actions b with $a_i = b_i$ we have $s \xrightarrow{b} t$. That is, the set of states reachable in a single transition from s depends only on the action performed by agent i . In this case, the transition relation can be presented more succinctly as a subset of $S \times (\cup_{i \in Ags} A_i) \times S$.

A *strategy* for agent $i \in Ags$ in such an environment E is a function $\alpha : S \rightarrow \mathcal{P}(Acts_i) \setminus \{\emptyset\}$, selecting a set of actions of the agent at each state.¹ We call these the actions *enabled* at the state. A *group strategy*, or *strategy profile*, for a group G is a tuple $\alpha_G = \langle \alpha_i \rangle_{i \in G}$ where each α_i is a strategy for agent i . A strategy α_i is *deterministic* if $\alpha_i(s)$ is a singleton for all s . A strategy α_i for agent i is *uniform* if for all states s, t , if $O_i(s) = O_i(t)$, then $\alpha_i(s) = \alpha_i(t)$. A strategy $\alpha_G = \langle \alpha_i \rangle_{i \in G}$ for a group G is *locally uniform (deterministic)* if α_i is uniform (respectively, deterministic) for each agent $i \in G$.² Given an environment E , we write $\Sigma^{det}(E)$ for the set of deterministic strategies, $\Sigma^{unif}(E)$ for the set of all locally uniform joint strategies, and $\Sigma^{unif, det}(E)$ for the set of all deterministic locally uniform joint strategies.

We now define an interpreted system that contains all the possible runs generated when agents Ags behave by choosing a strategy from some set Σ of joint strategies in the context of an environment E . To enable reference to the strategy being used by agent $i \in Ags$, we treat the strategy as a component of the global state, and introduce the notation “ $\sigma(i)$ ” as a name referring to agent i ’s strategy. For $G \subseteq Ags$, we write

¹More generally, a strategy could be a function of the history, but we focus here on strategies that depend only on the final state.

²We prefer the term “locally uniform” to just “uniform” in the case of groups, since we could say a strategy α for group G is *globally uniform* if for all states s, t , if $O_i(s) = O_i(t)$ for all $i \in G$, then $\alpha_i(s) = \alpha_i(t)$ for all $i \in G$. While we do not pursue this in the present paper, this notion would be interesting in settings where agents share information to collude on their choice of move.

$\sigma(G)$ for the set $\{\sigma(i) \mid i \in G\}$. Additionally, we include an agent e for representing the state of the environment.

Technically, $\sigma(i)$ will be treated as an agent in the context of temporal epistemic logic. This enables us to take the value of the local state of agent $\sigma(i)$ to be the strategy in use by agent i . We will permit use of the agents $\sigma(i)$ in epistemic operators. This provides a way to refer, using distributed knowledge operators D_G where G contains the new strategic agents $\sigma(i)$, to what agents would know, should they take into account not just their own observations, but also information about other agent's strategies. For example, the distributed knowledge operator $D_{\{i, \sigma(i)\}}$ captures the knowledge that agent i has, taking into account the strategy that it is running. $D_{\{i, \sigma(i), \sigma(j)\}}$ captures what agent i would know, taking into account its own strategy and the strategy being used by agent j . Various applications of the usefulness of this expressiveness are given in the sections on applications.

We note, however, that unlike the base agent $i \in \text{Ags}$, the agent $\sigma(i)$ is not one of the agents in the environment E , and it is not associated with any actions. The agent $\sigma(i)$ exists only in the interpreted system that we generate from E . (A similar remark applies to the special agent e , which is also not associated with any actions.)

Formally, given an environment $E = \langle S, I, \{Acts_i\}_{i \in \text{Ags}}, \rightarrow, \{O_i\}_{i \in \text{Ags}}, \pi \rangle$ for agents Ags , where $O_i : S \rightarrow L_i$ for each $i \in \text{Ags}$, and a set $\Sigma \subseteq \prod_{i \in \text{Ags}} \Sigma_i$ of joint strategies for the group Ags , we define the *strategy space* interpreted system $I(E, \Sigma) = (\mathcal{R}, \pi')$. The system $I(E, \Sigma)$ has global states $\mathcal{G} = S \times \prod_{i \in \text{Ags}} L_i \times \prod_{i \in \text{Ags}} \Sigma_i$. Intuitively, each global state consists of a state of the environment E , a local state for each agent i in E , and a strategy for each agent i . We index the components of this cartesian product by e , the elements of Ags and the elements of $\sigma(\text{Ags})$, respectively. We take the set of runs \mathcal{R} of $I(E, \Sigma)$ to be the set of all runs $r : \mathbb{N} \rightarrow \mathcal{G}$ satisfying the following constraints, for all $m \in \mathbb{N}$ and $i \in \text{Ags}$

1. $r_e(0) \in I$ and $\langle r_{\sigma(i)}(0) \rangle_{i \in \text{Ags}} \in \Sigma$,
2. $r_i(m) = O_i(r_e(m))$,
3. $(r_e(m), a, r_e(m+1)) \in \rightarrow$ for some joint action $a \in Acts$ such that for all $j \in \text{Ags}$ we have $a_j \in \alpha_j(r_j(m))$, where $\alpha_j = r_{\sigma(j)}(m)$, and
4. $r_{\sigma(i)}(m+1) = r_{\sigma(i)}(m)$.

The first constraint, intuitively, says that runs start at an initial state of E , and the initial strategy profile at time 0 is one of the profiles in Σ . The second constraint states that the agent i 's local state at time m is the observation that agent i makes of the state of the environment at time m . The third constraint says that evolution of the state of the environment is determined at each moment of time by agents choosing an action by applying their strategy at that time to their local state at that time. The joint action resulting from these individual choices is then resolved into a transition on the state of the environment using the transition relation from E . The final constraint says that agents' strategies are fixed during the course of a run. Intuitively, each agent picks a strategy, and then sticks to it. The interpretation π' of $I(E, \Sigma)$ is determined from the interpretation π of E by taking $\pi'(r, m) = \pi(r_e(m))$ for all points (r, m) .

Our epistemic strategy logic is now just an instantiation of the extended temporal epistemic logic in the strategy space generated by an environment. That is, we start with an environment E and an associated set of strategies Σ , and then work with the language $\text{ETLK}(Ags \cup \sigma(Ags) \cup \{e\}, Prop, Var)$ in the interpreted system $I(E, \Sigma)$. We call this instance of the language $\text{ESL}(Ags, Prop, Var)$, or just ESL when the parameters are implicit.

We will be interested in a number of fragments of ESL that turn out to have lower complexity. We define $\text{ESL}^-(Ags, Prop, Var)$, or just ESL^- , to be the language $\text{ECTL}(Ags \cup \sigma(Ags) \cup \{e\}, Prop, Var)$ in the interpreted system $I(E, \Sigma)$. Another fragment of the language that will be of interest is the language $\text{CTL}^*\text{K}(Ags \cup \sigma(Ags) \cup \{e\}, Prop, Var)$ (again interpreted in a system $I(E, \Sigma)$) in which we omit the constructs \exists and $e_i(x)$; this is a standard branching time temporal epistemic language except that it contains the special agents $\sigma(Ags)$.

3 Model Checking

Model checking is the problem of computing whether a formula of a logic holds in a given model. We now consider the problem of model checking ESL and various of its fragments.

Since interpreted systems are always infinite objects, we use environments to give a finite input for the model checking problem. For an environment E , a set of strategies Σ for E , and a context Γ for $I(E, \Sigma)$, we write $\Gamma, E, \Sigma \models \phi$ if $\Gamma, I(E, \Sigma), (r, 0) \models \phi$ for all runs r of $I(E, \Sigma)$. Often, the formula ϕ will be a sentence, i.e., will have all variables x in the scope of an operator $\exists x$. In this case the statement $\Gamma, E, \Sigma \models \phi$ is independent of Γ and we write simply $E, \Sigma \models \phi$. The model checking problem is to determine whether $\Gamma, E, \Sigma \models \phi$ for a finite state environment E , a set Σ of strategies and a context Γ , where ϕ is an ESL formula.

For generality, we abstract Σ to a parameterized class such that for each environment E , the set $\Sigma(E)$ is a set of strategies for E . When C is a complexity class, we say that the parameterized class Σ *can be presented in* C , if the problem of determining, given an environment E and a joint strategy α for E , whether $\alpha \in \Sigma(E)$, is in complexity class C . For example, the class $\Sigma(E)$ of all strategies for E can be PTIME -presented, as can $\Sigma^{\text{unif}}(E)$, $\Sigma^{\text{det}}(E)$ and $\Sigma^{\text{unif}, \text{det}}(E)$.

We first consider the complexity of model checking the full language ESL . The following result gives an upper bound of EXPSPACE for this problem.

Theorem 1 *Let Σ be a parameterized class of strategies for environments E that can be presented in EXPSPACE (or any lower class). The complexity of deciding, given an environment E , an ESL formula ϕ and a context Γ for the free variables in an ESL formula ϕ relative to E and $\Sigma(E)$, whether $\Gamma, E, \Sigma(E) \models \phi$, is in EXPSPACE .*

Proof: The problem can be reduced to that of model checking the temporal epistemic logic CTL^*K obtained by omitting the constructs \exists and $e_i(x)$ from the language ESL . This is known to be PSPACE -complete.³ The reduction involved involves an ex-

³The result is stated explicitly in [16], but techniques sufficient for a proof (involving guessing a labelling

ponential blowup of size of both the formula and the environment, so we obtain an EXPSPACE upper bound.

Model checking for temporal epistemic logic takes as input a formula and a structure that is like an environment, except that its transitions are not based on a set of actions for the agents. More precisely, an *epistemic transition system* for a set of agents Ags is a tuple $\mathcal{E} = \langle S, I, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$, where S is a set of states, $I \subseteq S$ is the set of initial states, $\rightarrow \subseteq S \times S$ is a state transition relation, for each $i \in Ags$, component $O_i : S \rightarrow L_i$ is a function giving an observation in some set L_i for the agent i at each state, and $\pi : S \rightarrow \mathcal{P}(Prop)$ is a propositional assignment. A *run* of \mathcal{E} is a sequence $r : \mathbb{N} \rightarrow S$ such that $r(0) \in I$ and $r(k) \rightarrow r(k+1)$ for all $k \in \mathbb{N}$. To ensure that every partial run can be completed to a run, we assume that the transition relation is *serial*, i.e., that for all states s there exists a state t such that $s \rightarrow t$.

Given an epistemic transition system \mathcal{E} , we define an interpreted system $I(\mathcal{E}) = (\mathcal{R}, \pi')$ as follows. For a run $r : \mathbb{N} \rightarrow S$ of \mathcal{E} , define the lifted run $\hat{r} : \mathbb{N} \rightarrow S \times \prod_{i \in Ags} L_i$ (here $L_e = S$), by $\hat{r}_e(m) = r(m)$ and $\hat{r}_i(m) = O_i(r(m))$ for $i \in Ags$. Then we take \mathcal{R} to be the set of lifted runs \hat{r} with r a run of \mathcal{E} . The assignment π' is given by $\pi'(r, m) = \pi(r(m))$. The model checking problem for temporal epistemic logic is to decide, given an epistemic transition system \mathcal{E} and a formula ϕ , whether $I(\mathcal{E}), (r, 0) \models \phi$ for all runs r of $I(\mathcal{E})$.

Given an environment $E = \langle S, I, Acts, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$ for language $ESL(Ags, Prop, Var)$, we first introduce a set of new propositions $Prop^* = \{p_{(s, \alpha)} \mid s \in S, \alpha \in \Sigma(E)\}$. We then define the epistemic transition system $\mathcal{E} = \langle S^*, I^*, \rightarrow^*, \{O_i^*\}_{i \in Ags}, \pi^* \rangle$ for the language $ETLK(Ags \cup \sigma(Ags), Prop \cup Prop^*, Var)$ extended by the propositions $Prop^*$, as follows:

1. $S^* = S \times \Sigma(E)$, i.e., a state of \mathcal{E} consists of a state of E together with a joint strategy,
2. $I^* = I \times \Sigma(E)$,
3. $(s, \alpha) \rightarrow^* (t, \beta)$ iff $s \xrightarrow{a} t$ for some joint action a such that $a \in \alpha(s)$, and $\beta = \alpha$,
4. $O_i^*(s, \alpha) = O_i(s)$,
5. $\pi^*(s, \alpha) = \pi(s) \cup \{p_{(s, \alpha)}\}$.

We can treat the states $(s, \alpha) \in S^*$ as tuples indexed by $Ags \cup \sigma(Ags) \cup \{e\}$ by taking $(s, \alpha)_i = O_i(s)$ and $(s, \alpha)_{\sigma(i)} = \alpha_i$ for $i \in Ags$, and $(s, \alpha)_e = s$. Note that a joint strategy for an environment E can be represented in space $|S| \times |Acts|$. Thus, the size of \mathcal{E} is $O(2^{poly(|E|)})$. Note also that the construction of \mathcal{E} can be done in EXPSPACE so long as verifying whether an individual strategy α is in $\Sigma(E)$ can be done in EXPSPACE.

We also need a transformation of the formula. Given a formula ϕ of ESL and a context Γ for E , we define a formula ϕ^Γ , inductively, by

1. $p^\Gamma = p$, for $p \in Prop$,

of states by knowledge subformulas in order to reduce the problem to LTL model checking and also verifying the guess by LTL model checking were already present in [50]. The branching operator A can be treated as a knowledge operator for purposes of the proof.

2. $e_i(x)^\Gamma = \bigvee_{g \in S^*, g_i = \Gamma(x)_i} p_g$
3. $(\neg\phi)^\Gamma = \neg\phi^\Gamma, (\phi_1 \wedge \phi_2)^\Gamma = \phi_1^\Gamma \wedge \phi_2^\Gamma,$
4. $(\bigcirc\phi)^\Gamma = \bigcirc(\phi^\Gamma), (\phi_1 U \phi_2)^\Gamma = (\phi_1^\Gamma)U(\phi_2^\Gamma),$
5. $\exists x(\phi)^\Gamma = \bigvee_{g \in S^*} \phi^{\Gamma[g/x]}.$

Plainly the size of ϕ^Γ is $O(2^{\text{poly}(|E|, |\phi|)})$, and this formula is in $\text{CTL}^*K(\text{Ags} \cup \sigma(\text{Ags}), \text{Prop} \cup \text{Prop}^*)$. A straightforward inductive argument based on the semantics shows that $\Gamma, E, \Sigma(E) \models \phi$ iff $I(E) \models \phi^\Gamma$. It therefore follows from the fact that model checking CTL^*K with respect to the observational semantics for knowledge is in PSPACE that ESL model checking is in EXPSpace. \square

The following result shows that a restricted version of the model checking problem, where we consider systems with just one agent and uniform deterministic strategies is already EXPSpace hard.

Theorem 2 *The problem of deciding, given an environment E for a single agent, and an ESL sentence ϕ , whether $E, \Sigma^{\text{unif}}(E) \models \phi$, is EXPSpace-hard.*

Proof: We show how polynomial size inputs to the problem can simulate exponential space deterministic Turing machine computations. Let $T = \langle Q, q_0, q_f, A_I, A_T, \delta \rangle$ be a one-tape Turing machine solving an EXPSpace-complete problem, with states Q , initial state q_0 , final (accepting) state q_f , input alphabet A_I , tape alphabet $A_T \supseteq A_I$, and transition function $\delta : Q \times A_T \rightarrow Q \times A_T \times \{L, R\}$. We assume that T runs in space $2^{p(n)} - 2$ for a polynomial $p(n)$, and that the transition relation is defined so that the machine idles in its final state on accepting, and idles in some other state on rejecting. The tape alphabet A_T is assumed to contain the blank symbol \perp .

Define $C_{T,Q} = A_T \cup (A_T \times Q)$ to be the set of “cell-symbols” of T . We may represent a configuration of T as a finite sequence over the set $C_{T,Q}$, containing exactly one element (x, q) of $A_T \times Q$, representing a cell containing symbol x where the machine’s head is positioned, with the machine in state q . For technical reasons, we pad configurations with a blank symbol to the left and right (so configurations take space $2^{p(n)}$), so that the initial configuration has the head at the second tape cell and, without loss of generality, assume that the machine is designed so that it never moves the head to the initial padding blank. This means that the transition function δ can also be represented as a set of tuples $\delta^* \subseteq C_{T,Q}^4$, such that $(a, b, c, d) \in \delta^*$ iff, whenever the machine is in a configuration with a, b, c at cells at positions $k-1, k, k+1$, respectively, the next configuration has d at the cell at position k .

Given the TM T and a number $N = p(n)$ we construct an environment $E_{T,N}$ such that for every input word w , with $|w| = n$, there exists a sentence ϕ_w of size polynomial in n such that $E_{T,N}, \Sigma^{\text{unif}}(E_{T,N}) \models \phi_w$ iff T accepts w . The idea of the simulation is to represent a run of the Turing machine, using space 2^N , by representing the sequence of configurations of T for the computation consecutively along a run r of the environment $E_{T,N}$. Each cell of a configuration will be encoded as a block of $N+1$ consecutive moments of time in r . Not all runs of $E_{T,N}$ will correctly encode a computation of the

machine, so we use the formula to check whether a run of T has been correctly encoded in a given run of E_T .

The environment E has propositions $C_{T,Q} \cup \{c\} \cup \{t_0, \dots, t_{N-1}\}$. Propositions from $C_{T,Q}$ are used to represent cell elements, and c is used to represent the bits of a counter that indicates the position of the cell being represented. In particular, a cell in a configuration, at position $b_{N-1} \dots b_0$, in binary, and containing symbol $a \in C_{T,Q}$, will be represented by a sequence of $N+1$ states, the first of which satisfies proposition a , such that for $i = 0 \dots N-1$, element $i+2$ of the sequence satisfies c iff $b_i = 1$.

We take the set of states of the environment to be

$$S = \{s_x \mid x \in C_{T,Q}\} \cup \{c_0, c_1\} \cup \{(t_i, j) \mid i = 0 \dots N-1, j \in \{0, 1\}\}.$$

The set of initial states of the environment is defined to be $I = \{s_\perp\} \cup \{(t_0, 0), \dots, (t_{N-1}, 0)\}$. We define the assignment π so that $\pi(s_a) = \{a\}$ for $a \in A_{T,Q}$, $\pi(c_0) = \emptyset$, $\pi(c_1) = \{c\}$ and $\pi((t_i, 0)) = \{t_i\}$ and $\pi((t_i, 1)) = \{t_i\} \cup \{c\}$.

We take the set of actions of the single agent to be the set $\{a_0, a_1\}$. The transition relation \rightarrow is defined so that the only transitions are

$$\begin{aligned} s_x &\xrightarrow{a_k} c_i && \text{for } x \in C_{T,Q} \text{ and } i \in \{0, 1\}, \\ c_i &\xrightarrow{a_k} c_j && \text{for } i, j \in \{0, 1\}, \\ c_i &\xrightarrow{a_k} s_x && \text{for } x \in C_{T,Q} \text{ and } i \in \{0, 1\}, \\ (t_i, j) &\xrightarrow{a_k} (t_i, k) && \text{for } i = 0 \dots N-1. \end{aligned}$$

Intuitively, this forces the runs starting at state s_\perp to alternate between selecting a symbol from $C_{T,Q}$ and a sequence of bits $\{0, 1\}$ for the counter. The states of the form (t_i, j) for $j \in \{0, 1\}$ each form an isolated component in the transition relation, and are used to ensure that there is a sufficiently rich set of strategy choices for strategies to encode counter values.

The length of the counter sequence segments of a run generated by this transition system can vary within the run, but we can use a formula of length $O(N^2)$ to state that these segments always have length N wherever they appear in the run; let ϕ_{clock}^N be the formula

$$\Box(\alpha_{T,Q} \Rightarrow (\bigcirc^{N+1} \alpha_{T,Q} \wedge \bigwedge_{i=1 \dots N} \bigcirc^i \neg \alpha_{T,Q}))$$

where we write $\alpha_{T,Q}$ for $\bigvee_{x \in C_{T,Q}} x$. By definition of the transition relation, this formula holds on a run starting in state s_\perp just when it consists of states of the form s_x alternating with sequences of states of the form c_i of length exactly N .

The transition system generates arbitrary such sequences of states c_i of length N , intuitively constituting a guess for the correct counter value. Note that a temporal formula of length $O(N^2)$ can say that these guesses for the counter values are correct, in that the counter values encoded along the run are $0, 1, 2, \dots, 2^N - 1, 0, 1, 2, \dots, 2^N - 1$ (etc). Specifically, this is achieved by the following formula ϕ_{count}^N :

$$\phi_{zero} \wedge \Box \left(\alpha_{T,Q} \Rightarrow \left(\begin{aligned} &(\phi_{max} \Rightarrow \bigcirc^{N+1}(\phi_{zero})) \wedge \\ &\bigwedge_{i=1 \dots N} ((\bigcirc^i c \wedge \dots \bigcirc^{i-1} c \wedge \bigcirc^i \neg c) \Rightarrow \\ &\quad \bigcirc^{N+1}(\bigcirc^i \neg c \wedge \dots \bigcirc^{i-1} \neg c \wedge \bigcirc^i c) \wedge \bigwedge_{j=i+1 \dots N} ((\bigcirc^j c) \Leftrightarrow (\bigcirc^{j+N+1} c))) \end{aligned} \right) \right)$$

where $\phi_{zero} = \bigwedge_{i=1 \dots N} \bigcirc^i \neg c$ and $\phi_{max} = \bigwedge_{i=1 \dots N} \bigcirc^i c$.

The following formula ϕ_{init}^w then says that the run is initialized with word $w = a_1 \dots a_n$

$$\perp \wedge \bigcirc^{N+1}((q_0, a_1) \wedge \bigcirc^{N+1}(a_2 \wedge \bigcirc^{N+1}(\dots \bigcirc^{N+1}(a_n \wedge \bigcirc((\alpha_{T,Q} \Rightarrow \perp) U (\alpha_{T,Q} \wedge \phi_{zero}))) \dots)))$$

where \perp is the blank symbol. This formula has size $O(N \cdot |w|) = O(p(n) \cdot n)$. (Intuitively, the formula says that the sequence of symbols w is followed by a sequence of \perp symbols up to some time that the counter has value 0: this could be an occurrence of counter 0 that lies further than the start of the second configuration. However, adding the next formula we discuss will prevent incorrect blanks in the second configuration and force the until condition to be satisfied at the first position of the second configuration.

We now need a formula that expresses that whenever we consider two consecutive configurations C, C' encoded in a run, C' is derived from C by a single step of the TM T . The padding blanks are easily handled by the following formula ϕ_{pad} :

$$\Box((\alpha_{T,Q} \wedge (\phi_{zero} \vee \phi_{max})) \Rightarrow \perp)$$

For the remaining cell positions, we need to express that for each cell position $k = 1 \dots 2^N - 2$, the cell value at position k in C' is determined from the cell value at positions $k-1, k, k+1$ in C according to the transition relation encoding δ^* . This means that we need to be able to identify the corresponding positions k in C and C' . To capture the counter value at a given position in the run, we represent counter values using a strategy for the single agent, as follows.

We define the observation function O_1 for the single agent in $E_{T,N}$, so that $O_1(t_i) = i$ for $i = 0 \dots N-1$. (The values of the observation function on other states are not used in the encoding, and can be defined arbitrarily.) The number with binary representation $B = b_{N-1} \dots b_0$ can then be represented by the strategy α_B such that $\alpha_B(i) = a_{b_i}$, for $i = 0 \dots N-1$. Comparing this representation with the encoding of numbers along runs, the following formula $\phi_{num}(x)$ expresses that the number encoded at the present position in the run is the same as the number encoded in the strategy of agent 1 in the global state denoted by variable x :

$$\alpha_{T,Q} \wedge \bigwedge_{i=0 \dots N-1} (\bigcirc^{i+1} c) \Leftrightarrow \neg D_{\emptyset} \neg (e_{\sigma(1)}(x) \wedge t_i \wedge \bigcirc c)$$

Note that, by the definition of the transition system and the observation function, the value of $\bigcirc c$ at a state where t_i holds encodes whether the strategy selects a_0 or a_1 on observation $i = 0 \dots N-1$. We may now check that the transitions of the TM are correctly computed along the run by means of the following formula ϕ_{trans} :

$$\Box \left(\bigwedge_{(a,b,c,d) \in \delta^*} (a \wedge \neg \phi_{max} \wedge \bigcirc^{N+1}(b \wedge \neg \phi_{max} \bigcirc^{N+1}(c))) \Rightarrow \bigcirc^{N+1} \exists x [\phi_{num}(x) \wedge \bigcirc((\neg \phi_{num}(x)) U (\phi_{num}(x) \wedge d))] \right)$$

Intuitively, here x captures the number encoded at the cell containing the symbol b , and the U operator is used to find the next occurrence in the run of this number. The occurrences of ϕ_{max} ensure that the three positions considered in the formula do not span across a boundary between two configurations.

To express that the machine accepts we just need to assert that the accepting state is reached; this is done by the formula $\phi_{accept} = \diamond q_f$.

Combining these pieces, we get that the TM accepts input w just when

$$E_{T,N} \models (\phi_{clock}^N \wedge \phi_{count}^N \wedge \phi_{init}^w \wedge \phi_{trans}) \Rightarrow \phi_{accept}$$

holds, i.e., when every run that correctly encodes a computation of the machine is accepting.

□

Combining Theorem 1 and Theorem 2 we obtain the following characterization of the complexity of ESL model checking.

Corollary 1 *Let $\Sigma(E)$ be a PTIME presented class of strategies for environments E . The complexity of deciding, given an environment E , an ESL formula ϕ and a context Γ for the free variables in an ESL formula ϕ relative to E and $\Sigma(E)$, whether $\Gamma, E, \Sigma(E) \models \phi$, is EXPSPACE-complete.*

The high complexity for ESL model checking motivates the consideration of fragments that have lower model checking complexity. We demonstrate two orthogonal fragments for which the complexity of model checking is in a lower complexity class. One is the fragment ESL^- , where we allow the operators $\exists x.\phi$ and $e_i(x)$, but restrict the use of the temporal operators to be those of the branching time temporal logic CTL. In this case, we have the following result:

Theorem 3 *Let Σ be a PSPACE-presented class of strategies. The problem of deciding, given an environment E , a formula ϕ of ESL^- , and a context Γ for the free variables of ϕ relative to E and $\Sigma(E)$, whether $\Gamma, E, \Sigma(E) \models \phi$, is in PSPACE.*

Proof: We observe that the following fact follows straightforwardly from the semantics for formulas ϕ of ESL^- : for a context Γ for the free variables of ϕ relative to E and $\Sigma(E)$, and for two points (r, n) and (r', n') of $I(E, \Sigma(E))$ with $r(n) = r'(n')$, we have that $\Gamma, I(E, \Sigma(E)), (r, n) \models \phi$ iff $\Gamma, I(E, \Sigma(E)), (r', n') \models \phi$. That is, satisfaction of formula relative to a context at a point depends only on the global state at the point, and not on other details of the run containing the point. For a global state (s, α) of $I(E, \Sigma(E))$, define the boolean $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ to be TRUE just when $\Gamma, I(E, \Sigma(E)), (r, n) \models \phi$ holds for some point (r, n) of $I(E, \Sigma(E))$ with $r(n) = (s, \alpha)$. By the above observation, we have that $\Gamma, E, \Sigma(E) \models \phi$ iff $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ holds for all initial states s of E and all strategies $\alpha \in \Sigma(E)$. Since we may check these conditions one at a time, strategies α can be represented in space linear in $|Env|$, and deciding $\alpha \in \Sigma(E)$ is in PSPACE, it suffices to show that $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$ is decidable in PSPACE.

We proceed by describing an APTIME algorithm for $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi)$, and using the fact that APTIME = PSPACE [10]. The algorithm operates recursively, with the following cases:

1. If $\phi = p$, for $p \in Prop$, then return TRUE if $p \in \pi(s)$, else return FALSE.

2. If $\phi = e_i(x)$, then return TRUE if $(s, \alpha)_i = \Gamma(x)_i$, else return FALSE.
3. If $\phi = \phi_1 \wedge \phi_2$, then universally call $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi_1)$ and $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi_2)$.
4. If $\phi = \neg\phi_1$, then return the complement of $SAT(\Gamma, E, \Sigma, (s, \alpha), \phi_1)$.
5. If $\phi = A \circ \phi_1$ then universally choose a state t such that $s \xrightarrow{a} t$ for some $a \in \alpha(t)$, and call $SAT(\Gamma, E, \Sigma, (t, \alpha), \phi_1)$. The other temporal operators from CTL are handled similarly. (In the case of operators using U , we need to run a search for a path through the set of states of E generated by the strategy α , but this is easily handled in APTIME.)
6. If $\phi = D_G \phi_1$, then universally choose a global state (t, β) such that $(s, \alpha) \sim_G^D (t, \beta)$ and universally
 - (a) decide if $\beta \in \Sigma(E)$, and
 - (b) call $REACH(t, \beta)$, and
 - (c) call $SAT(\Gamma, E, \Sigma, (t, \beta), \phi_1)$.

(Here $REACH(t, \beta)$ decides whether state t is reachable in E from some initial state when the agents run the joint strategy β ; this is trivially in PSPACE. Deciding $\beta \in \Sigma(E)$ is in PSPACE by the assumption that Σ is PSPACE-presented.)
7. If $\phi = C_G \phi_1$, then universally guess a global state (t, β) and universally do the following:
 - (a) Decide $(s, \alpha) \sim_G^C (t, \beta)$ using an existentially branching binary search for a path of length at most $|S| \times |\Sigma(E)|$. For all states (u, γ) on this path it should be verified that $REACH(u, \gamma)$ and that $\gamma \in \Sigma(E)$. The maximal length of the path is in the worst case exponential in $|E|$, but the binary search can handle this in APTIME.
 - (b) call $SAT(\Gamma, E, \Sigma, (t, \beta), \phi_1)$.
8. If $\phi = \exists x(\phi_1)$, then existentially guess a global state (t, β) , and universally
 - (a) decide if $\beta \in \Sigma(E)$, and
 - (b) call $REACH(t, \beta)$, and
 - (c) call $SAT(\Gamma[(t, \beta)/x], E, \Sigma, (s, \alpha), \phi_1)$.

A straightforward argument based on the semantics of the logic shows that the above correctly computes SAT.

We remark that a more efficient procedure for checking that $(s, \alpha) \sim_G^C (t, \beta)$ is possible in the typical case where Σ is a cartesian product of sets of strategies for each of the agents. In this case, if there exists a witness chain then there is one of length at most $|S|$. Let $G = G_1 \cup \sigma(G_2)$ such that $G_1, G_2 \subseteq \text{Ags}$. The number of steps through the relation $\cup_{i \in G} \sim_i$ required to witness $(s, \alpha) \sim_G^C (t, \beta)$ depends on the sets G_1, G_2 as follows:

1. If $G_1 = G_2 = \emptyset$ then we must have $(s, \alpha) = (t, \beta)$ and a chain of length 0 suffices.
2. If G_1 is nonempty and $G_2 = \emptyset$ then we must have $s (\cup_{i \in G_1} \sim_i)^* t$, but β can be arbitrary, and this component can be changed in any step. A path of length $|S|$ suffices in this case.
3. If $G_1 = \emptyset$ and $G_2 = \{i\}$ is a singleton, then we must have $\alpha_i = \beta_i$, but s and t can be arbitrary. A path of length one suffices in this case.
4. If $|G_1| \geq 1$, say $i \in G_1$, and $G_2 = \{j\}$ is a singleton, then $(\cup_{i \in G} \sim_i)^*$ is the universal relation and a path of length 2 suffices. In particular, for any $(s, \alpha), (t, \beta)$ we have $(s, \alpha) \sim_i (s, \beta) \sim_{\sigma(j)} (t, \beta)$.
5. If $|G_2| \geq 2$ then $(\cup_{i \in G} \sim_i)^*$ is the universal relation and a path of length 2 suffices. In particular, for any $(s, \alpha), (t, \beta)$ and $i, j \in G_2$ we have $(s, \alpha) \sim_{\sigma(i)} (s, \alpha') \sim_{\sigma(j)} (t, \beta)$, where $\alpha'_i = \alpha_i$ and $\alpha'_k = \beta_k$ for all $k \neq i$.

□

The following result shows that the PSPACE upper bound from this result is tight, already for formulas that use strategy agents in the CTLK operators, but make no direct uses of the constructs $\exists x$ and $e_i(x)$. (Recall that use of strategy agents in epistemic operators can be eliminated at the cost of introducing $\exists x$ and $e_i(x)$.)

Theorem 4 *The problem of deciding, given an environment E for two agents and a formula ϕ of $CTLK(Ags \cup \sigma(Ags), Prop)$, whether $E, \Sigma^{unif.det}(E) \models \phi$ is PSPACE hard.*

Proof: By reduction from the satisfiability of Quantified Boolean Formulae (QBF). An instance of QBF is a formula ϕ of form

$$Q_1 x_1 \dots Q_n x_n (\gamma)$$

where $Q_1, \dots, Q_n \in \{\exists, \forall\}$ and γ is a formula of propositional logic over propositions x_1, \dots, x_n . The QBF problem is to decide, given a QBF instance ϕ , whether it is true. We construct an environment E_ϕ and a formula ϕ^* of CTLK using strategic agents $\sigma(i)$ such that the QBF formula ϕ is true iff we have $E_\phi, \Sigma^{unif.det}(E_\phi) \models \phi^*$.

Given the QBF formula ϕ , we construct the environment $E_\phi = \langle S, I, \{Acts_i\}_{i \in Ags}, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$ for 2 agents $Ags = \{1, 2\}$ and propositions $Prop = \{p_0, \dots, p_n, q_1, a_2\}$ as follows.

1. The set of states $S = \{s_0\} \cup \{s_{t,j,k} \mid t \in \{1 \dots n\}, j, k \in \{0, 1\}\}$.
2. The set of initial states is $I = \{s_0\}$.
3. The actions of agent i are $A_i = \{0, 1\}$, for each $i \in Ags$.

4. The transition relation is defined, to consist of the following transitions, where $j, j', k, k' \in \{0, 1\}$

$$\begin{aligned} s_0 &\xrightarrow{(j', k')} s_{1, j', k'} \\ s_{t, j, k} &\xrightarrow{(j', k')} s_{t+1, j', k'} \quad \text{for } t = 1 \dots n-1 \\ s_{n, j, k} &\xrightarrow{(j', k')} s_{n, j, k} . \end{aligned}$$

5. Observations are defined so that $O_i(s_0) = 0$ and $O_i(s_{t, j, k}) = t$.

6. The assignment π is defined by $\pi(s_0) = \{p_0\}$, $\pi(st, j, k) = \{p_t\} \cup \{q_1 \mid j = 1\} \cup \{q_2 \mid k = 1\}$.

Intuitively, the model sets up $n + 1$ moments of time $t = 0 \dots, n + 1$. Both agents observe only the value of the moment of time, so that for each agent, a strategy selects an action 0 or 1 at each moment of time. We may therefore encode an assignment to the proposition variables $x_1 \dots x_n$ by the actions chosen by an agent at times $0, \dots, n - 1$. The action chosen by each agent at time $t \in \{0 \dots n - 1\}$ is recorded in the indices of the state at time $t + 1$, i.e. if the state at time $t + 1$ is $s_{t+1, j, k}$ then agent 1 chose action j at time t , and agent 2 chose action k .

We work with two agents, each of whose strategies is able to encode an assignment, in order to alternate between the two encodings. At each step, one of the strategies is assumed to encode an assignment to the variables x_1, \dots, x_m . This strategy is fixed, and we universally or existentially guess the other strategy in order to obtain a new value for the variable x_{m+1} . We then check that the guess has maintained the values of the existing assignment to x_1, \dots, x_m by comparing the two strategies.

More precisely, let $val_i(x_j)$ be the formula $K_{\{\sigma(i)\}}(p_{j-1} \Rightarrow EX(q_i))$ for $i = 1, 2$ and $j = 1 \dots n$. This states that at the current state, the strategy of agent i selects action 1 at time $j - 1$, so it encodes an assignment making x_j true. For $m = 1 \dots n$, let $agree(m)$ be the formula

$$\bigwedge_{j=1 \dots m} D_{\{\sigma(1), \sigma(2)\}}(p_{j-1} \Rightarrow (EX(q_1) \Leftrightarrow EX(q_2)))$$

This says that the assignments encoded by the strategies of the two agents agree on the values of the variables $x_1 \dots, x_m$. Assuming, without loss of generality, that n is even, and that the quantifier sequence in ϕ is $(\exists \forall)^{n/2}$, given the QBF formula ϕ , define the formula ϕ^* to be

$$\begin{aligned} \neg D_0 \neg (D_{\{\sigma(1)\}}(agree(1) \Rightarrow \\ \neg D_{\{\sigma(2)\}} \neg (agree(2) \wedge \\ D_{\{\sigma(1)\}}(agree(3) \Rightarrow \\ \neg D_{\{\sigma(2)\}} \neg (agree(4) \wedge \dots \\ \vdots \\ D_{\{\sigma(1)\}}(agree(m-1) \Rightarrow \gamma^+) \dots)) \end{aligned}$$

where γ^+ is the formula obtained by replacing each occurrence of a variable x_j in γ by the formula $val_2(x_j)$. Intuitively, the first operator $\neg D_0 \neg$ existentially chooses a value for variable x_1 , encoded in $\sigma(1)$, the next operator $D_{\{\sigma(1)\}}$ remembers this strategy

while encoding a universal choice of value for variable x_2 in $\sigma(2)$, and the formula $agree(1)$ checks that the existing choice for x_1 is preserved in $\sigma(2)$. Continued alternation between the two strategies adds universal or existential choices for variable values while preserving previous choices. It can then be shown that the QBF formula ϕ is true iff $E_\phi, \Sigma^{unif, det} \models \phi^*$. \square

Combining Theorem 3 and Theorem 4, we obtain the following:

Corollary 2 *Let Σ be a PSPACE-presented class of strategies. The problem of deciding if $\Gamma, E, \Sigma(E) \models \phi$, given an environment E , a formula ϕ of ESL^- and a context Γ for the free variables of ϕ relative to E and $\Sigma(E)$, is PSPACE complete.*

Since PSPACE is strictly contained in EXPSpace, this result shows a strict improvement in complexity as a result of the restriction to the CTL-based fragment. We remark that, by a trivial generalization of the standard state labelling algorithm for model checking CTL to handle the knowledge operators, the problem of model checking the logic $CTLK(Ags, Prop)$ in the systems $\mathcal{I}(\mathcal{E})$ generated by an epistemic transitions system \mathcal{E} is in PTIME. Thus, there is a jump in complexity from CTLK as a result of the move to the strategic setting, even without the addition of the operators $\exists x.\phi$ and $e_i(x)$. However, this jump is not so large as the jump to the full logic ESL.

An orthogonal restriction of ESL is to retain the CTL^* temporal basis, i.e., to allow full use of LTL operators, but to add only epistemic operators and strategy agents, but omit use of the operators $\exists x.\phi$ and $e_i(x)$. This gives the logic $CTL^*K(Ags \cup \sigma(Ags), Prop)$. For this logic we also see an improvement in the complexity of model checking compared to full ESL, as is shown in the following result.

Theorem 5 *Let $\Sigma(E)$ be a PSPACE presented class of strategies for environments E . The complexity of deciding, given an environment E , and a CTL^*K formula ϕ for agents $\{e\} \cup Ags(E) \cup \sigma(Ags(E))$, whether $E, \Sigma(E) \models \phi$, is PSPACE-complete.*

Proof: The lower bound is straightforward from the fact that linear time temporal logic LTL is a sublanguage of CTL^*K , and model checking LTL is already PSPACE-hard [49]. For the upper bound, we describe an alternating PTIME algorithm, and invoke the fact that $APTIME = PSPACE$ [10]. We abbreviate $\mathcal{I}(E, \Sigma(E))$ to \mathcal{I} .

For a formula ϕ , write $\maxk(\phi)$ for the maximal epistemic subformulas of ϕ , defined to be the set of subformulas of the form $A\psi$ or $C_G\phi$ or $D_G\psi$ for some set G of basic and strategic agents, which are themselves not a subformula of a larger subformula of ϕ of one of these forms. Note that $A\psi$ can be taken to be epistemic because it is equivalent to $D_{\{e\} \cup \sigma(Ags)}\psi$; in the following we assume that $A\psi$ is written in this form. Also note that for epistemic formulas ψ , satisfaction at a point depends only on the global state, i.e., for all points (r, m) and (r', m') of \mathcal{I} , we have that if $r(m) = r'(m')$ then $\mathcal{I}, (r, m) \models \psi$ iff $\mathcal{I}, (r', m') \models \psi$. Thus, for global states (s, α) of \mathcal{I} , we may write $\mathcal{I}, (s, \alpha) \models \psi$ to mean that $\mathcal{I}, (r, m) \models \psi$ for some point (r, m) with $r(m) = (s, \alpha)$.

Define a ϕ -labelling of E to be a mapping $L : S \times \maxk(\phi) \rightarrow \{0, 1\}$, giving a truth value for each maximal epistemic subformula of ϕ . A ϕ -labelling can be represented in space $|S| \times |\phi|$. Note that if we treat the maximal epistemic subformulas of ϕ as if

they were atomic propositions, evaluated at the states of E using the ϕ -labelling L , then ϕ becomes an LTL formula, evaluable on any path in E with respect to the labelling L . Verifying that all α -paths from a state s satisfy ϕ with respect to L is then exactly the problem of LTL model checking, for which there exists an APTIME procedure $\text{ASAT}(E, (s, \alpha), L, \phi)$. For this to correspond to model checking in \mathcal{I} , we require that L gives the correct answers for the truth value of the formula at each state (s, α) , i.e., that $L(\psi) = 1$ iff $\mathcal{I}, (s, \alpha) \models \psi$. We handle this by means of a guess and verify technique.

To handle the verification, we define an alternating PTIME algorithm $\text{KSAT}(E, \Sigma, (s, \alpha), \phi)$ for ϕ an epistemic formula such that $\text{KSAT}(E, \Sigma, (s, \alpha), \phi)$ returns TRUE iff $\mathcal{I}, (s, \alpha) \models \phi$. The definition is recursive and uses a call to the procedure ASAT . Specifically, $\text{KSAT}(E, \Sigma, (s, \alpha), D_G\phi)$ operates as follows:

1. universally guess a state t of E and a joint strategy β in E , then
2. verify that t is reachable in E using joint strategy β , that $(s, \alpha) \sim_G (t, \beta)$, and that β is in $\Sigma(E)$, then
3. existentially guess a ϕ -labelling L of E , then
4. universally,
 - (a) call $\text{ASAT}(E, (t, \beta), L, \phi)$, and
 - (b) for each state w and formula $\psi \in \text{maxk}(\phi)$, call $\text{KSAT}(E, \Sigma, (w, \beta), \psi)$.

Note that step 4(b) verifies that the ϕ -labelling L is correct.

For $\text{KSAT}(E, \Sigma, (s, \alpha), C_G\phi)$, the procedure is similar, except that instead of verifying that $(s, \alpha) \sim_G (t, \beta)$ in the second step, we need to verify that $(s, \alpha) (\cup_{i \in G} \sim_i)^* (t, \beta)$. This is easily handled in APTIME by a standard recursive procedure that guesses a midpoint of the path and branching universally to verify the existence of the left and right halves of the chain. (See the proof of Theorem 3 for some further discussion on this point.)

To solve the model checking problem in \mathcal{I} , we can now apply the following alternating procedure:

1. universally guess a global state (s, α) of \mathcal{I} , then branch existentially to the following cases:
 - (a) if s is an initial state of E return FALSE, else return TRUE,
 - (b) if $\alpha \in \Sigma(E)$, return FALSE, else return TRUE,
 - (c) call $\text{KSAT}(E, \Sigma, (s, \alpha), A\phi)$.

Evidently, each of the alternating procedures runs in polynomial time internally, and the number of recursive calls is $O(|\phi|)$. It follows that the entire computation is in AP-TIME = PSPACE. \square

It is interesting to note that, although $\text{CTL}^*\text{K}(\text{Ags} \cup \sigma(\text{Ags}))$ is significantly richer than the temporal logic LTL, the added expressiveness comes without an increase in complexity: model checking LTL is already PSPACE-complete [49].

4 Applications

We now consider a range of applications of the logic ESL. Since the full logic has a highly complex model checking problem, we endeavor to use restricted fragments of the logic with lower complexity whenever possible.

4.1 Connections to variants of ATEL

Alternating temporal logic (ATL) [1] is a generalization of the branching time temporal logic CTL that can express the capability of agent strategies to bring about temporal effects. Essentially, each branching construct $A\phi$ is generalized to an *alternating* construct $\langle\langle G \rangle\rangle\phi$ for a group G of agents, where ϕ is a “prefix temporal” formula such as $X\phi'$, $F\phi'$, $G\phi'$ or $\phi_1 U \phi_2$, as would be used to construct a CTL operator. Intuitively, $\langle\langle G \rangle\rangle\phi$ says that the group G has a strategy for ensuring that ϕ holds, irrespective of what the other agents do.

Alternating temporal epistemic logic (ATEL), adds epistemic operators to ATL [26]. As a number of subtleties arise in the formulation of such logics, several variants of ATEL have since been developed. In this section, we consider a number of such variants and argue that our framework is able to express the main strategic concepts from these variants. We begin by recalling ATEL as defined in [26]. Various modellings of the environments in which agents operate have been used in the literature; we base our modelling on the notion of environment introduced above.

The syntax of ATEL is given as follows:

$$\phi \equiv p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \langle\langle G \rangle\rangle\bigcirc\phi \mid \langle\langle G \rangle\rangle\Box\phi \mid \langle\langle G \rangle\rangle(\phi_1 U \phi_2) \mid K_i\phi \mid D_G\phi \mid C_G\phi$$

where $p \in Prop$, $i \in Ags$ and $G \subseteq Ags$. As usual, we may define $E_G\phi$ as $\bigwedge_{i \in G} K_i\phi$. The intuitive meaning of the constructs is as in CTL*K above, with additionally $\langle\langle G \rangle\rangle\phi$ having the intuitive reading that group G has a strategy for assuring that ϕ holds. The semantics is given by a relation $E, s \models^\Sigma \phi$, where $E = \langle S, I, Acts, \rightarrow, \{O_i\}_{i \in Ags}, \pi \rangle$ is an environment, $s \in S$ is a state of E , and ϕ is a formula. For reasons discussed below, we parameterize the definition on a set Σ of strategies for groups of agents in the environment E . For the definition, we need the notion of a path in E : this is a function $\rho : \mathbb{N} \rightarrow S$ such that for all $k \in \mathbb{N}$ there exists a joint action a with $(\rho(k), a, \rho(k+1)) \in \rightarrow$. A path ρ is *from* a state s if $\rho(0) = s$. A path ρ is *consistent* with a strategy α for a group G if for all $k \in \mathbb{N}$ there exists a joint action a such that $(\rho(k), a, \rho(k+1)) \in \rightarrow$ and $a_i \in \alpha_i(\rho(k))$ for all $i \in G$.

The relation $E, s \models^\Sigma \phi$ is defined inductively on the structure of the formula ϕ :

- $E, s \models^\Sigma p$ if $p \in \pi(s)$;
- $E, s \models^\Sigma \neg\phi$ if not $E, s \models^\Sigma \phi$;
- $E, s \models^\Sigma \phi \wedge \psi$ if $E, s \models^\Sigma \phi$ and $E, s \models^\Sigma \psi$;
- $E, s \models^\Sigma \langle\langle G \rangle\rangle\bigcirc\phi$ if there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ from s that are consistent with α_G , we have $E, \rho(1) \models^\Sigma \phi$;

- $E, s \models^\Sigma \langle\langle G \rangle\rangle \Box \phi$ if there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ from s that are consistent with α_G , we have $E, \rho(k) \models^\Sigma \phi$ for all $k \in \mathbb{N}$;
- $E, s \models^\Sigma \langle\langle G \rangle\rangle (\phi U \psi)$ if there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ from s that are consistent with α_G , there exists $m \geq 0$ such that $E, \rho(m) \models^\Sigma \psi$, and for all $k < m$, we have $E, \rho(k) \models^\Sigma \phi$.
- $E, s \models^\Sigma K_i \phi$ if $E, t \models^\Sigma \phi$ for for all $t \in S$ with $t \sim_i s$;
- $E, s \models^\Sigma D_G \phi$ if $E, t \models^\Sigma \phi$, for all $t \in S$ with $(s, t) \in \bigcap_{i \in G} \sim_i$.
- $E, s \models^\Sigma C_G \phi$ if $E, t \models^\Sigma \phi$ for for all $t \in S$ with $(s, t) \in (\bigcup_{i \in G} \sim_i)^*$;

The specific version of ATEL defined in [26] is obtained from the above definitions by taking $\Sigma = \Sigma^{det} = \{\sigma_G \mid G \subseteq \text{Ags}, \sigma_G \text{ a deterministic } G\text{-strategy in } E\}$. That is, following the definitions for ATL, this version works with arbitrary deterministic group strategies, in which an agent selects its action as if it has full information of the state. This aspect of the definition has been critiqued by Jonker [35] and (in the case of ATL without epistemic operators) by Schobbens [47], who argue that this choice is not in the spirit of the epistemic extension, in which observations are intended precisely to represent that agents do not have full information of the state. They propose that the definition instead be based on the set $\Sigma^{det, unif} = \{\sigma_G \mid G \subseteq \text{Ags}, \sigma_G \text{ a locally uniform deterministic } G\text{-strategy in } E\}$. This ensures that in choosing an action, agents are able to use only the information available in their observations.

We concur that the use of locally uniform strategies is the more appropriate choice, but in either event, we now argue that our approach using strategy space is able to express everything that can be expressed in ATEL. Consider the following translation from ATEL to CTL^{*}K($\text{Prop}, \text{Ags} \cup \sigma(\text{Ags}) \cup \{e\}$). For a formula ϕ , we write ϕ^* for the translation of ϕ , defined inductively on the construction of ϕ by the following rules

$$\begin{aligned}
p^* &= p & (\neg \phi)^* &= \neg \phi^* & (\phi_1 \wedge \phi_2)^* &= \phi_1^* \wedge \phi_2^* \\
(K_i \phi)^* &= K_i \phi^* & (C_G \phi)^* &= C_G \phi^* \\
(\langle\langle G \rangle\rangle \bigcirc \phi)^* &= \neg K_e \neg D_{\{e\} \cup \sigma(G)} \bigcirc \phi^* \\
(\langle\langle G \rangle\rangle \Box \phi)^* &= \neg K_e \neg D_{\{e\} \cup \sigma(G)} \Box \phi^* \\
(\langle\langle G \rangle\rangle \phi_1 U \phi_2)^* &= \neg K_e \neg D_{\{e\} \cup \sigma(G)} (\phi_1^* U \phi_2^*)
\end{aligned}$$

Given a strategy $\alpha = \langle \alpha_i \rangle_{i \in G}$ for a group of agents G in an environment E , define the *completion* of the strategy to be the joint strategy $\text{comp}(\alpha) = \langle \alpha'_i \rangle_{i \in G}$ with $\alpha'_i = \alpha_i$ for $i \in G$ and with $\alpha'_i(s) = \text{Acts}_i$ for all $i \in \text{Ags} \setminus G$ and $s \in S$. Intuitively, this operation completes the group strategy to a joint strategy for all agents, by adding the random strategy for all agents not in G . Given a set of strategies Σ for groups of agents, we define $\text{comp}(\Sigma) = \{\text{comp}(\alpha) \mid \alpha \in \Sigma\}$. Say that a set Σ of group strategies is *restrictable* if for every $\alpha \in \Sigma$ for group of agents G and every group $G' \subseteq G$, the restriction $\alpha_{G'}$ of α to agents in G' is also in Σ . Say that Σ is *extendable* if for every strategy for a group $G' \subseteq G$, there exists a strategy $\alpha' \in \Sigma$ for group G whose restriction $\alpha'_{G'}$ to G' is equal to α . For example, the set of all group strategies, and the set of all locally uniform group strategies, are both restrictable and extendable.

For an environment E , write $E[S/I]$ for the environment obtained by making all states in E be initial, i.e., replacing the set of initial states I of E by the set of all states S of E . (This is a technical transformation that we require because E may have temporally unreachable states that would not occur in an interpreted system constructed from E , but that can be accessed via an equivalence relation \sim_i in the semantics of ATEL.) We can then show the following.

Theorem 6 *For every environment E , nonempty set of group strategies Σ that is restrictable and extendable, for every state s of E and ATEL formula ϕ , we have $E, s \models^\Sigma \phi$ iff for all (equivalently, some) points (r, m) of $\mathcal{I}(E[S/I], \text{comp}(\Sigma))$ with $r_e(m) = s$ we have $\mathcal{I}(E[S/I], \text{comp}(\Sigma)), (r, m) \models \phi^*$.*

Proof: For brevity, we write just \mathcal{I} for $\mathcal{I}(E[S/I], \text{comp}(\Sigma))$. For the claim that the quantifiers “for all” and “some” are interchangeable in the right hand side, note that formulas of the form ϕ^* are boolean combinations of atomic formulas and formulas of the form $K_i\psi$ and $K_e\psi$, whose semantics at a point (r, m) depends only on $r_e(m)$. (Recall that, by construction, $r_e(m) = r'_e(m')$ implies $r_i(m) = r'_i(m')$.) This gives the implication from the “some” case to the “for all” case. For the implication from the “for all” case to the “some” case, note that the “for all” case is never trivial because for all states s of E , there exists a point (r, m) of \mathcal{I} with $r_e(m) = s$. This follows from the fact that all states are initial in $E[S/I]$ and that the transition relation is serial, so that any group strategy α in Σ is consistent with an infinite path from initial state s . This corresponds to a run r with $r(0) = (s, \text{comp}(\alpha))$.

It therefore suffices to show that $E, s \models^\Sigma \phi$ iff for all points (r, m) of \mathcal{I} with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \phi^*$. We proceed by induction on the construction of ϕ . The base case of atomic propositions, as well as the cases for the boolean constructs, are trivial.

Consider $\phi = K_i\psi$. Then $(K_i\phi)^* = K_i\phi^*$. We suppose first that $E, s \models^\Sigma \phi$ and show that for all points (r, m) of \mathcal{I} with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \phi^*$, i.e., $\mathcal{I}, (r, m) \models K_i\psi^*$. Let (r, m) be a point of \mathcal{I} with $r_e(m) = s$. We need to show that for all points (r', m') of \mathcal{I} with $(r, m) \sim_i (r', m')$ we have $\mathcal{I}, (r', m') \models \psi^*$. But if $(r, m) \sim_i (r', m')$ then $r'_e(m') \sim_i r_e(m) = s$ in E . Thus, from $E, s \models^\Sigma K_i\psi$ it follows that $E, r'_e(m') \models^\Sigma \psi$. By the induction hypothesis, we obtain that $\mathcal{I}, (r', m') \models \psi^*$, as required.

Conversely, suppose that for all points (r, m) of \mathcal{I} with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models K_i\psi^*$. We show that $E, s \models^\Sigma K_i\psi$. Let t be any state of E with $s \sim_i t$. We have to show $E, t \models^\Sigma \psi$. First, since s is an initial state of $E[S/I]$, there exists a run r of \mathcal{I} with $r_e(0) = s$, and joint strategy equal to any strategy in $\text{comp}(\Sigma)$, so we take $m = 0$, and we have $\mathcal{I}, (r, m) \models K_i\psi^*$. Then for all points (r', m') of \mathcal{I} with $r'_e(m') = t$, we have $(r, 0) \sim_i (r', m')$, from which it follows that $\mathcal{I}, (r', m') \models \psi^*$. By the induction hypothesis, we have $E, t \models \psi$, as required. This completes the proof for the case of $\phi = K_i\psi$. The argument for the distributed and common knowledge operators is similar, and left to the reader.

We consider next the case of $\phi = \langle\langle G \rangle\rangle \psi$. (The arguments for the formulas $\langle\langle G \rangle\rangle \Box \psi$ and $\langle\langle G \rangle\rangle \psi_1 U \psi_2$ are analogous and left to the reader.) We show that $E, s \models^\Sigma \langle\langle G \rangle\rangle \psi$ iff for all points (r, m) of \mathcal{I} with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\} \cup \sigma(G)} \Box \phi^*$. Suppose first $E, s \models^\Sigma \langle\langle G \rangle\rangle \psi$. Then there exists a strategy $\alpha_G \in \Sigma$ for group G such that for all paths ρ of E from s that are consistent with α_G we have $E, \rho(1) \models^\Sigma \phi$. Let $\alpha = \text{comp}(\alpha_G)$

(note that this is in $\text{comp}(\Sigma)$) and let r' be a run of \mathcal{I} with $r'(0) = (s, \sigma)$. Let (r, m) be a point of \mathcal{I} with $r_e(m) = s$. We show that $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$. Indeed, because $(r, m) \sim_e (r', m')$, it suffices to show that $\mathcal{I}, (r', m') \models D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$. For this, suppose that (r'', m'') is a point of \mathcal{I} with $(r', m') \sim_{\{e\} \cup \sigma(G)} (r'', m'')$. Then $r''(m'') = (t, \alpha')$ implies that $\alpha'_i = \alpha_i$ for all $i \in G$. Thus, the path $\rho = r''_e(m'')r''_e(m'' + 1) \dots$ in E is consistent with α_G , and $\rho(0) = r''_e(m'') = r'_e(m') = r_e(m) = s$. It follows that $E, \rho(1) \models^\Sigma \psi$. Using the induction hypothesis, it follows that $\mathcal{I}, (r'', m'' + 1) \models \phi^*$, hence $\mathcal{I}, (r', m') \models \bigcirc \phi^*$. This completes the argument that $\mathcal{I}, (r', m') \models D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$.

Conversely, suppose that for all points (r, m) of \mathcal{I} with $r_e(m) = s$ we have $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$. We show that $E, s \models^\Sigma \langle\langle G \rangle\rangle \bigcirc \psi$. Applying the existence argument above, let (r, m) be a point of \mathcal{I} with $r_e(m) = s$, and hence $\mathcal{I}, (r, m) \models \neg K_e \neg D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$. Then there exists a point (r', m') of \mathcal{I} such that $(r', m') \sim_e (r, m)$ and $\mathcal{I}, (r', m') \models D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$. Let $r'(m') = (t, \alpha)$. Then in fact $t = s$, and we have $\alpha \in \text{comp}(\Sigma)$. By definition, there exists a strategy $\alpha'_{G'} \in \Sigma$ for some set of agents G' such that $\alpha = \text{comp}(\alpha'_{G'})$. This means that for any agent $i \in G \cap G'$ we have that $\alpha_i = (\alpha'_{G'})_i$, and for any agent $i \in G \setminus G'$ we have that α_i is the random strategy. Let α''_G be a strategy in Σ for group G such that $(\alpha''_G)_{G \cap G'} = (\alpha'_{G'})_{G \cap G'}$. Such a strategy must exist, by restrictability and extendability of Σ , and we also have $(\alpha''_G)_{G \cap G'} = \alpha_{G \cap G'}$.

To prove that $E, s \models^\Sigma \langle\langle G \rangle\rangle \bigcirc \psi$, we show that for every path ρ of E from s consistent with α''_G , we have $E, \rho(1) \models^\Sigma \psi$. For this, let ρ be a path from s consistent with α''_G . We claim that ρ is also consistent with $\text{comp}(\alpha_{G \cap G'})$. For agents in $G \cap G'$, this follows from the fact that $(\alpha''_G)_{G \cap G'} = \alpha_{G \cap G'}$. For all other agents this holds because $\text{comp}(\alpha_{G \cap G'})$ is the random strategy. Since $\alpha_{G \cap G'} = (\alpha'_{G'})_{G \cap G'}$ is a restriction of a strategy in Σ , it is itself a strategy in Σ , so $\text{comp}(\alpha_{G \cap G'})$ is a strategy in $\text{comp}(\Sigma)$. Since s is an initial state of $E[S/I]$, there exists a run r'' of \mathcal{I} with $r''(0) = (s, \text{comp}(\alpha_{G \cap G'}))$ and $r''_e[0 \dots \infty] = \rho$. The strategies α and $\text{comp}(\alpha_{G \cap G'})$ are both random for agents in $G \setminus G'$ and identical by definition for agents in $G \cap G'$. Hence $(r', m') \sim_{\{e\} \cup \sigma(G)} (r'', 0)$. Thus, we obtain from $\mathcal{I}, (r', m') \models D_{\{e\} \cup \sigma(G)} \bigcirc \phi^*$ that $\mathcal{I}, (r'', 0) \models \bigcirc \phi^*$, and hence $\mathcal{I}, (r'', 1) \models \psi^*$. By the induction hypothesis, and noting that $r''_e(1) = \rho(1)$, we obtain that $E, \rho(1) \models^\Sigma \psi$. \square

We remark that our translation maps ATEL into the fragment $\text{CTLK}(Ags \cup \sigma(Ags) \cup \{e\}, \text{Prop})$ that we have shown to have PSPACE-complete model checking complexity. This strongly suggests that this fragment has a strictly stronger expressive power than ATEL, since the complexity of model checking ATEL logic, assuming uniform strategies, is known to be P^{NP} -complete. (The class P^{NP} consists of problems solvable by PTIME computations with access to an NP oracle.) For ATEL, model checking can be done with a polynomial time (with respect to the size of formula) computation with access to an oracle that is in NP with respect to both the number of states and the number of joint actions. In particular, [47] proves this upper bound and [33] proves a matching lower bound.

Similar translation results can be given for other alternating temporal epistemic logics from the literature. We sketch a few of these translations here.

Jamroga and van der Hoek [34] note that ATEL admits situations consisting of an environment E and a state s where $E, s \models K_i \langle\langle i \rangle\rangle \phi$, i.e., in every state consistent with agent i 's knowledge, some strategy for agent i is guaranteed to satisfy ϕ , but still there

is no strategy for agent i that agent i knows will work to achieve ϕ . They formulate a construct $\langle\langle G \rangle\rangle_{\mathcal{K}(H)}^\bullet \phi$ that says, effectively, that there is a strategy for a group G that another group H knows (for notion of group knowledge \mathcal{K} , which could be E for everyone knows, D for distributed knowledge, or C for common knowledge) to achieve goal ϕ . More precisely,

$$E, s \models \langle\langle G \rangle\rangle_{\mathcal{K}(H)}^\bullet \phi \quad \text{if there exists a uniform strategy } \alpha \text{ for group } G \text{ such that for all states } t \text{ with } s \sim_H^\mathcal{K} t, \text{ we have that all paths } \rho \text{ from } t \text{ that are consistent with } \alpha \text{ satisfy } \phi.$$

Here $\sim_H^\mathcal{K}$ is the appropriate epistemic indistinguishability relation on states of E . The particular case $\langle\langle G \rangle\rangle_{E(G)}^\bullet \phi$ is also proposed as the semantics for the ATL construct $\langle\langle G \rangle\rangle \phi$ in [47, 35, 31].

The construct $\langle\langle G \rangle\rangle_{D(H)}^\bullet \phi$ can be represented in the $\text{CTLK}(Ags \cup \sigma(Ags) \cup \{e\}, Prop)$ fragment of ESL as

$$\neg K_e \neg D_{H \cup \sigma(G)} \phi.$$

Intuitively, here the first modal operator $\neg D_e \neg$ switches the strategy of all the agents while maintaining the state s , thereby selecting a strategy α for group G in particular, and the next operator $D_{\{H, \sigma(G)\}}$ verifies that the group H knows that the strategy G being used by group G guarantees ϕ . Similarly, $\langle\langle G \rangle\rangle_{E(H)}^\bullet \phi$ can be represented as

$$\neg K_e \neg \bigwedge_{i \in H} D_{\{i\} \cup \sigma(G)} \phi.$$

In the case of the operator $\langle\langle H \rangle\rangle_{C(G)}^\bullet \phi$, the definition involves the common knowledge that a group G of agents would have if they were to reason taking into consideration the strategy being used by another group H . This does not appear to be expressible using $\text{CTL}^*K(Ags \cup \sigma(Ags) \cup \{e\}, Prop)$. In particular, the formula $C_{G \cup \sigma(H)} \phi$ does not give the intended meaning. Instead, what needs to be expressed is the greatest fixpoint X of the equation $X \equiv \bigwedge_{i \in G} D_{\{i\} \cup \sigma(H)}(X \wedge \phi)$. The language $\text{CTL}^*K(Ags \cup \sigma(Ags), Prop)$ does not include fixpoint operators and it does not seem that the intended meaning is expressible. On the other hand, it can be expressed with ESL in a natural way by the formula $C_G(e_{\sigma(H)}(x) \Rightarrow \phi)$, which says that it is common knowledge to the group G that ϕ holds if the group H is running the strategy profile captured by the variable x . Using this idea, the construct $\langle\langle H \rangle\rangle_{C(G)}^\bullet \phi$ can be represented with ESL as

$$\exists x. C_G(e_{\sigma(H)}(x) \Rightarrow \phi).$$

This gives a reduction of these complex operators of [34] to a set of standard epistemic operators. (We remark that a carefully stated equivalence result requires an appropriate treatment of initial states in the environment E , similar to that used in the treatment of ATEL above.)

An alternate approach to decomposing the operators $\langle\langle G \rangle\rangle_{\mathcal{K}(H)}^\bullet$ is proposed in [31]. By comparison with our standard approach to the semantics of the epistemic operators, this proposal uses “constructive knowledge” operators which require a nonstandard semantics in which formulas are evaluated at sets of states rather than at individual states. The constructive strategic logic CSL [31] has a set of constructive knowledge

operators $\{\mathcal{D}, \mathcal{E}, C\}$, and is shown in [32] to have a normal form: every subformula starting with a constructive knowledge operator \mathcal{K}_G is of the form $\mathcal{K}_{G_1} \dots \mathcal{K}_{G_n} \phi$ where ϕ starts with a strategy modality and $\mathcal{K} \in \{\mathcal{D}, \mathcal{E}, C\}$. A normal form CSL formula $\mathcal{K}_{G_1} \dots \mathcal{K}_{G_n} \langle\langle H \rangle\rangle \phi$ can be represented in ESL as

$$\exists x. K_{G_1} \dots K_{G_n} (e_{\sigma(H)}(x) \Rightarrow \phi)$$

where K_{G_i} is the usual knowledge operator corresponding to \mathcal{K}_{G_i} , for all $i \in \{1..n\}$.

Another work by van der Hoek, Jamroga and Wooldridge [51] introduces constants that refer to strategies, and adds to ATL a new (counterfactual) modality $C_i(c, \phi)$, with the intended reading “if it were the case that agent i committed to the strategy denoted by c , then ϕ ”. The formula ϕ here is not permitted to contain further references to agent i strategies. To interpret the formula $C_i(c, \phi)$ in an environment E , the environment is first updated to a new environment E' by removing all transitions that are inconsistent with agent i running the strategy referred to by c , and then the formula ϕ is evaluated in E' . After introducing propositional constants $p_{i,c}$ that say that agent i is running the strategy referred to by c , the formula $C_i(c, \phi)$ could be expressed in our framework as $D_{\{e\} \cup \sigma(Ags \setminus \{i\})}(p_{i,c} \Rightarrow \phi^{+\sigma(i)})$ where in the translation $\phi^{+\sigma(i)}$ of ϕ we ensure that there is no further deviation from the strategy of agent i by adding $\sigma(i)$ to the group of every knowledge operator occurring later in the translation.

The logic of [51] does not include epistemic operators. Were they to be added in the obvious way, the effect would be to make the meaning of the knowledge operator (with respect to the agent’s knowledge concerning the strategies in play) dependent on the context in the formula. In particular, in $(K_j \phi) \wedge C_i(c, K_j \psi)$, the first occurrence of K_j would refer to an agent j who does not know that i is playing c , whereas the second K_j would effectively refer to our $K_{\{j, \sigma(i)\}}$, i.e., j ’s knowledge taking into account that i is playing c . Our framework is more expressive in that we can continue to use the former meaning even in the second context.

4.2 Game Theoretic Solution Concepts

It has been shown for a number of logics for strategic reasoning that they are expressive enough to state a variety of game theoretic solution concepts, e.g., [51, 11] show that Nash Equilibrium is expressible. We now sketch the main ideas required to show that the fragment $\text{CTLK}(Ags \cup \sigma(Ags) \cup \{e\}, \{Prop\})$ of our framework also has this expressive power. We assume two players $Ags = \{0, 1\}$ in a normal form game, and assume that these agents play a deterministic strategy. The results in this section can be easily generalized to multiple players and extensive form games.

Given a game \mathcal{G} we construct an environment $E_{\mathcal{G}}$ that represents the game. Each player has a set of actions that correspond to the moves that the player can make. We assume that $E_{\mathcal{G}}$ is constructed to model the game so that play happens in the first step from a unique initial state, and that subsequent transitions do not change the state.

Let u_i for $i \in \{0, 1\}$ be a variable denoting the utility gained by player i when play is finished. Let V_i be the set of possible values for u_i . We write $-i$ to denote the adversary of player i . We use formula

$$U_i(v) = \bigcirc (u_i = v)$$

to express that value v is player i 's utility once play finishes.

Nash equilibrium (NE) is a solution concept that states no player can gain by unilaterally changing their strategy. We may write

$$BR_i(v) = U_i(v) \wedge K_{\sigma(-i)} \bigwedge_{v' \in V_i} (U_i(v') \Rightarrow v' \leq v)$$

to express that, given the current adversary strategy, the value v attained by player i 's current strategy is the best possible utility attainable by player i , i.e., the present strategy of player i is a best response to the adversary. Thus

$$BR_i = \bigvee_{v \in V_i} BR_i(v)$$

says that player i is playing a best-response to the adversary's strategy. The following statement then expresses the existence of a Nash equilibrium for the game \mathcal{G} :

$$E_{\mathcal{G}}, \Sigma^{unif, det}(E_{\mathcal{G}}) \models \neg D_0 \neg (BR_0 \wedge BR_1).$$

That is, in a Nash equilibrium, each player is playing a best response to the other's strategy.

Perfect cooperative equilibrium (PCE) is a solution concept intended to overcome deficiencies of Nash equilibrium for explaining cooperative behaviour [24]. It says that each player does at least as well as she would if the other player were best-responding. The following formula

$$BU_i(v) = D_0 \left(\bigwedge_{v' \in V_i} ((BR_{-i} \wedge U_i(v')) \Rightarrow v' \leq v) \right)$$

states that v is as good as any utility that i can obtain if the adversary always best-responds to whatever i plays. Thus,

$$BU_i = \bigvee_{v \in V_i} (U_i(v) \wedge BU_i(v))$$

says that i is currently getting a utility as good the best utility that i can obtain if the adversary is a best-responder. Now, the following formula expresses the existence of perfect cooperative equilibrium for the game \mathcal{G} :

$$E_{\mathcal{G}}, \Sigma^{unif, det}(E_{\mathcal{G}}) \models \neg D_0 \neg (BU_0 \wedge BU_1)$$

That is, in a PCE, no player has an incentive to change their strategy, on the assumption that the adversary will best-respond to any change.

4.3 Reasoning about Knowledge-Based Programs

Knowledge-based programs [18] are a form of specification of a multi-agent system in the form of a program structure that specifies how an agent's actions are related

to its knowledge. They have been shown to be a useful abstraction for several areas of application, including the development of optimal protocols for distributed systems [18], robot motion planning [5], and game theoretic reasoning [22].

Knowledge-based programs cannot be directly executed, since there is a circularity in their semantics: which actions are performed depends on what the agents know, which in turn depends on which actions the agents perform. The circularity is not vicious, and can be resolved by means of a fixed point semantics, but it means that a knowledge-based program may have multiple distinct implementations (or none), and the problem of reasoning about these implementations is quite subtle. In this section, we show that our framework can capture reasoning about the set of possible implementations of a knowledge-based program.

We consider joint knowledge-based programs P (as defined by [18]) where for each agent i we have a knowledge-based program

$$P_i = \mathbf{do} \ \phi_1^i \rightarrow a_1^i \ [] \ \dots [] \ \phi_{n_i}^i \rightarrow a_{n_i}^i \ \mathbf{od}$$

where each ϕ_j^i is a formula of $\text{CTL}^*K(\text{Ags}, \text{Prop})$ of the form $K_i\psi$, and each a_i appears just once.⁴ Intuitively, this program says to repeat forever the following operation: non-deterministically execute one of the actions a_j^i such that the corresponding guard ϕ_j^i is true. To ensure that it is always the case that at least one action enabled, we assume that $\phi_1^i \vee \dots \vee \phi_{n_i}^i$ is a valid formula; this can always be ensured by taking the last condition $\phi_{n_i}^i$ to be the “otherwise” condition $K_i\neg(\phi_1^i \vee \dots \vee \phi_{n_i-1}^i)$, which is equivalent to $\neg(\phi_1^i \vee \dots \vee \phi_{n_i-1}^i)$ by introspection.

We present a formulation of semantics for knowledge-based programs that refactors the definitions of [18], following the approach of [37] which uses the notion of environment defined above rather than the original notion of *context*. A *potential implementation* of a knowledge-based program P in an environment E is a joint strategy α in E . Given a potential implementation α in E , we can construct the interpreted system $I(E, \{\alpha\})$, which captures the possible runs of E when the agents choose their actions according to the single possible joint strategy α . Given this interpreted system, we can now interpret the epistemic guards in P . Say that a state s of E is α -reachable if there is a point (r, m) of $I(E, \{\alpha\})$ with $r_e(m) = s$. We note that for a formula $K_i\phi$, and a point (r, m) of $I(E, \{\alpha\})$, the statement $I(E, \{\alpha\}), (r, m) \models K_i\phi$ depends only on the state $r_e(m)$ of the environment at (r, m) . For an α -reachable state s of E , it therefore makes sense to define satisfaction of $K_i\phi$ at s rather than at a point, by $I(E, \{\alpha\}), s \models K_i\phi$ if $I(E, \{\alpha\}), (r, m) \models K_i\phi$ for all (r, m) with $r_e(m) = s$. We define α to be an *implementation* of P in E if for all α -reachable states s of E and agents i , we have

$$\alpha_i(s) = \{a_j^i \mid 1 \leq j \leq n_i, \ I(E, \{\alpha\}), s \models \phi_j^i\}.$$

Intuitively, the right hand side of this equation is the set of actions that are enabled at s by P_i when the tests for knowledge are interpreted using the system obtained by

⁴The guards in [18] are allowed to be boolean combinations of formulas $K_i\psi$ and propositions p local to the agent: since for such propositions $p \Leftrightarrow K_i p$, and the operator K_i satisfies positive and negative introspection, our form for the guards is equally general. They do not require that a_i appears just once, but the program can always be put into this form by aggregating clauses for a_i into one and taking the disjunction of the guards.

running the strategy α itself. The condition states that the strategy is an implementation if it enables precisely this set of actions at every reachable state.

We now show that our framework for strategic reasoning can express the same content as a knowledge-based program by means of a formula, and that this enables the framework to be used for reasoning about knowledge-based program implementations.

We need one constraint on the environment. Say that an environment E is *action-recording* if for each $a \in Acts_i$ there exists an atomic proposition $did_i(a)$ such that for $s \in I$ we have $did_i(a) \notin \pi(s)$ and for all states s, t and joint actions a such that $(s, a, t) \in \rightarrow$, we have $did_i(b) \in \pi(t)$ iff $b = a_i$, for all agents i . It is easily seen that any environment can be made action-recording, just by adding a component to the states that records the latest joint action.

We can now express knowledge-based program implementations as follows. The main issue that we need to deal with is that the semantics of knowledge formulas in knowledge-based programs is given with respect to a system $I(E, \{\alpha\})$, in which it is *common knowledge* that the joint strategy in use is α . In general, strategies are not common knowledge in the strategy space $I(E, \Sigma)$ within which we wish to reason about knowledge-based program implementations. We handle this by means of a transformation of formulas.

For a formula ψ of $CTL^*K(Ags, Prop)$, write $\phi^\$$ for the formula of ESL obtained from the following recursively defined transformation:

$$\begin{aligned} p^\$ &= p & (\neg\phi)^\$ &= \neg\phi^\$ & (\phi_1 \wedge \phi_2)^\$ &= \phi_1^\$ \wedge \phi_2^\$ \\ (K_i\phi)^\$ &= D_{\{i\} \cup \sigma(Ags)} \phi^\$ \\ (D_G\phi)^\$ &= D_{G \cup \sigma(Ags)} \phi^\$ \\ (C_G\phi)^\$ &= \exists x (e_{\sigma(Ags)}(x) \wedge C_G(e_{\sigma(Ags)}(x) \Rightarrow \phi^\$) \\ A\phi^\$ &= A\phi^\$ \\ \bigcirc \phi^\$ &= \bigcirc \phi^\$ \\ (\phi_1 U \phi_2)^\$ &= (\phi_1^\$ U \phi_2^\$) \end{aligned}$$

Intuitively, this substitution says that knowledge operators in ϕ are to be interpreted as if it is known that the current joint strategy is being played. In the case of an operator K_i or, more generally, D_G , the translation handles this by adding $\sigma(Ags)$ to the set of agents that are kept fixed when moving through the indistinguishability relation. In the case of the common knowledge operator C_G , this does not work, for the same reasons as already explained in the context of the translation for the operator $\langle\langle H \rangle\rangle_{C(G)}^\bullet \phi$ in the section on ATEL above. Here we resolve this problem in a similar way using the operators $\exists x$ and $e_{Ags}(x)$.

Let

$$\mathbf{imp}(P) = D_{\sigma(Ags)} \left(\bigwedge_{i \in Ags, j=1 \dots n_i} ((\phi_j^i)^\$ \Leftrightarrow EX did_i(a_j^i)) \right).$$

Intuitively, this formula says that the current joint strategy gives an implementation of the knowledge-based program P . More precisely, we have the following:

Proposition 1 Suppose that P is a knowledge-based program. Let α be a locally uniform joint strategy in E and let r be a run of $\mathcal{I}^{unif}(E)$, in which the agents are running joint strategy α , i.e., $r(0) = (s, \alpha)$ for some state s . Let $m \in \mathbb{N}$. Then the strategy α is an implementation of knowledge-based program P in E iff $\mathcal{I}^{unif}(E), (r, m) \models \mathbf{imp}(P)$.

Note that here we check $\mathbf{imp}(P)$ in the system $\mathcal{I}^{unif}(E)$ that contains *all* joint strategies, not just a single strategy α , as in the definition of implementation for a knowledge-based program. The point of this is that *a priori*, we do not have an implementation at hand.

In particular, as a consequence of this result, it follows that several properties of knowledge-based programs (that do not make use of common knowledge operators) can be expressed in the system $\mathcal{I}^{unif}(E)$:

1. The statement that there exists an implementation of P in E can be expressed by

$$\mathcal{I}^{unif}(E) \models \neg D_0 \neg \mathbf{imp}(P)$$

2. The statement that all implementations of P in E guarantee that formula ϕ of $\text{CTL}^*K(\text{Ags}, \text{Prop})$ (which may contain knowledge operators) holds at all times can be expressed by

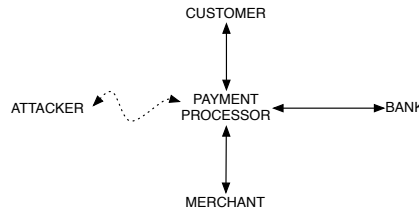
$$\mathcal{I}^{unif}(E) \models D_0(\mathbf{imp}(P) \Rightarrow \phi^S)$$

We remark that as a consequence of these encodings and the fact that the model checking problem described below is in PSPACE, we obtain that testing for the existence of an implementation of a knowledge-based program is in PSPACE, as is the problem of determining whether all implementations satisfy some formula. For testing existence, this result was known [18], but the result on verification has not previously been noted (though it could also have been shown using the techniques in [18].)

4.4 Computer Security Example: Erasure policies

Formal definitions of computer security frequently involve reference to the strategies available to the players, and to agent's reasoning based on these strategies. In this section we sketch an example that illustrates how our framework might be applied in this context.

Consider the scenario depicted in the following diagram:



A customer C can purchase items at a web merchant M . Payment is handled by a trusted payment processor P (this could be a service or device), which interacts with

the customer, merchant, and a bank B to securely process the payment. (To keep the example simple, we suppose that the customer and merchant use the same bank). One of the guarantees provided by the payment processor is to protect the customer from attacks on the customer's credit card by the merchant: the specification for the protocol that runs the transaction requires that the merchant should not obtain the customer's credit card number. In fact, the specification for the payment processor is that after the transaction has been successfully completed, the payment processor should *erase* the credit card data, to ensure that even the payment processor's state does not contain information about the customer's credit card number. The purpose of this constraint is to protect the customer against subsequent attacks by an attacker A , who may be able to use vulnerabilities in the payment processor's software to obtain access to the payment processor's state.

We sketch how one might use our framework to express the specification. To capture reasoning about all possible behaviours of the agents, and what they can deduce from knowledge of those behaviours, we work in $\mathcal{I}^{unif}(E)$ for a suitably defined environment E . To simplify matters, we take $\text{Ags} = \{C, M, P, A\}$. We exclude the strategy of the bank from consideration: this amounts to assuming that the bank has no actions and is trusted to run a fixed protocol. We similarly assume that the payment processor P has no actions, but in order to talk about what information is encoded in the payment processor's local state, we do allow that this agent has observations. The customer C may have actions such as entering the credit card number in a web form, pressing a button to submit the form to the payment processor, and pressing a button to approve or cancel the transaction. The customer observes variable cc , which records the credit card number drawn from a set CCN , and boolean variable $done$ which records whether the transaction is complete (which could mean either committed or aborted).

We assume that the attacker A has some set of exploit actions, as well as some innocuous actions (e.g., setting a local variable or performing *skip*). The effect of the exploit actions is to exploit a vulnerability in the payment processor's software and copy parts of the local state of the payment processor to variables that are observable by the attacker. We include in the environment state a boolean variable $exploited$, which records whether the attacker has executed an exploit action at some time in the past. The merchant M may have actions such as sending cost information to the payment processor and acknowledging a receipt certifying that payment has been approved by the bank (we suppose this receipt is transmitted from the bank to the merchant via the payment processor).

We may then capture the statement that the system is *potentially* vulnerable to an attack that exploits an erasure flaw in the implementation of the payment processor, by the following formula:

$$\neg D_0 \neg (\text{done} \wedge \bigvee_{x \in \text{CCN}} K_P(cc \neq x))$$

This says that there exist behaviours of the agents, which can (at least on some points in some runs) leave the payment processor in a state where the customer has received confirmation that the transaction is done, but in which the payment processor's local state somehow still encodes *some* information about the customer's credit card number. This encoding could be direct (e.g., by having a variable *customer_cc* that still stores

the credit card number) or indirect (e.g. by the local state including both a symmetric encryption key K and an encrypted version of the credit card number, $enc_customer_cc$, with value $\text{Encrypt}_K(cc)$ that was used for secure transmission to the bank). Note that for a breach of security, it is only required that the information suffices to *rule out* some credit card number (so that, e.g., knowing the first digit of the number would constitute a vulnerability)

The vulnerability captured by this formula is only potential, because it does not necessarily follow that the attacker is able to obtain the credit card information. Whether this is possible can be checked using the formula

$$\neg D_0 \neg (\text{done} \wedge \neg \text{exploited} \wedge EF \bigvee_{x \in \text{CCN}} D_{\{A, \sigma(A)\}}(cc \neq x)))$$

which says that it is possible for the attacker to obtain information about the credit card number even after the transaction is done. (To focus on erasure flaws, we deliberately wish to exclude here the possibility that the attack occurs during the processing of the transaction.) Note that here we assume that the attacker knows their own strategy when making deductions from the information obtained in the attack. This is necessary, because the attacker can typically write its own local variables, so it needs to be able to distinguish between a value it wrote itself and a value it copied from the payment processor.

However, even this formula may not be sufficiently strong. Suppose that the payment processor implements erasure by writing a random value to its variable $customer_cc$. Then, even if the attacker obtains a copy of this value, and it happens to be equal to the customer's actual credit card number, the attacker would not have any knowledge about the credit card number, since, as far as the attacker knows, it could be looking at a randomly assigned number. However, there may still be vulnerabilities in the system. Suppose that the implementation of the payment processor operates so that the customer's credit card data is not erased by randomization until the merchant has acknowledged the receipt of payment from the bank, but to avoid annoying the customer with a hanging transaction, the customer is advised that the transaction is approved (setting done true) if the merchant does not respond within a certain time limit. It is still the case that on observing the copied value of $customer_cc$, the attacker would not be able to deduce that this is the customer's credit card number, since it might be the result of erasure in the case that the merchant responded promptly. However, if the attacker knows that the merchant has not acknowledged the receipt, the attacker can then deduce that the value is not due to erasure. One way in which the attacker might know that the merchant has not acknowledged receipt is that the attacker is in collusion with the merchant, who has agreed to omit sending the required acknowledgement messages.

This type of attack can be captured by replacing the term $D_{\{A, \sigma(A)\}}(cc \neq x)$ by $D_{\{A, \sigma(A), \sigma(M)\}}(cc \neq x)$, capturing that the attacker reasons using knowledge of both its own strategy as well as the strategy of the merchant, or even $D_{\{A, \sigma(A), \sigma(M), M\}}(cc \neq x)$ for a collusion in which the merchant shares information observed. Similarly, to focus on erasure flaws in the implementation of the payment gateway, independently of the attackers capability, we would replace the term $K_P(cc \neq x)$ above by $D_{\{P, \sigma(M)\}}(cc \neq x)$.

We remark that in the case of the attacker’s knowledge, it would be appropriate to work with a perfect recall semantics of knowledge, but when using knowledge operators to express information in the payment gateway’s state for purposes of reasoning about erasure policy, the more appropriate semantics of knowledge is imperfect recall.

This example illustrates some of the subtleties that arise in the setting of reasoning about security and the way that our framework helps to represent them. Erasure policies have previously been studied in the computer security literature, beginning with [12], though generally without consideration of strategic behaviour by the adversary. However, many other notions in the security literature do involve reasoning about strategies and agent’s knowledge based on strategies, including nondeducibility on strategies [52] and robust declassification [53]. We leave the further exploration of such notions using our approach for future work.

5 Conclusion

We now discuss some related work and remark upon some questions for future research. The sections above have already made some references and comparisons to related work on each of the topics that we cover. Beside these references, the following are also worth mentioning.

A variant of propositional dynamic logic (PDL) for describing strategy profiles in normal form games subject to preference relations is introduced in [14]. This work does not cover temporal aspects as we have done in this paper. Another approach based on PDL is given in [44], which describes strategies by means of formulas.

A very rich generalization of ATEL for probabilistic environments is described in [46]. This proposal includes variables that refer to *strategy choices*, and strategic operators that may refer to these variables, so that statements of the form “when coalition A runs the strategy represented by variable S1, and coalition B runs the strategy represented by variable S2, and the remaining agents behave arbitrarily, then the probability that ϕ holds is at least δ ” can be expressed. Here a strategy choice maps each state, coalition and formula to a uniform imperfect recall strategy for the coalition. There are a number of syntactic restrictions compared to our logic. The epistemic operators in this approach apply only to state formulas rather than path formulas (in the sense of this distinction from CTL*.) Moreover, the strategic variables may be quantified, but only in the prefix of the formula. These constraints imply that notions such as “agent i knows that there exists a strategy by which it can achieve ϕ ” and “agent i knows that it has a winning response to every strategy chosen by agent j ” cannot be naturally expressed.

The extended temporal epistemic logic ETLK we have introduced, of which our epistemic strategy logic ESL is an instantiation with respect to a particular semantics, uses constructs that resemble constructs from *hybrid logic* [3]. Hybrid logic is an approach to the extension of modal logics that uses “nominals”, i.e., propositions p that hold at a single world. These can be used in combination with operators such as $\exists p$, which marks an arbitrary world as the unique world at which nominal p holds. Our construct $\exists x$ is closely related to the hybrid construct $\exists p$, but we work in a setting that is richer in both syntax and semantics than previous works. There have been a few works

using hybrid logic ideas in the context of epistemic logic [25, 45] but none are concerned with temporal logic. Hybrid temporal logic has seen a larger amount of study [4, 20, 19, 48], with variances in the semantics used for the model checking problem.

We note that if we were to view the variable x in our logic as a propositional constant, it would be true at a set of points in the system $I(E, \Sigma)$, hence not a nominal in that system. Results in [4], where a hybrid linear time temporal logic formula is checked in all paths in a given model, suggest that a variant of ESL in which x is treated as a nominal in $I(E, \Sigma)$ would have a complexity of model checking at least non-elementary, compared to our EXPSPACE and PSPACE complexity results.

Our PSPACE model checking result for $\text{CTLK}(Ags \cup \sigma(Ags))$ seems to be more closely related to the a result in [19] that model checking a logic $\text{HL}(\exists, @, F, A)$ is PSPACE-complete. Here F is essentially a branching time future operator and A is a universal operator (similar to our D_0), the construct $@_p\phi$ says that ϕ holds at the world marked by the nominal p , and $\exists p(\phi)$ says that ϕ holds after marking some world by p . The semantics in this case does not unfold the model into either a tree or a set of linear structures before checking the formula, so the semantics of the hybrid existential \exists is close to our idea of quantifying over global states. Our language, however, has a richer set of operators, even in the temporal dimension, and introduces the strategic dimension in the semantics. It would be an interesting question for future work to consider fragments of our language to obtain more a precise statement of the relationship with hybrid temporal logics.

Strategy Logic [11] is a (non-epistemic) generalization of ATL for perfect information strategies in which strategies may be explicitly named and quantified. Strategy logic has a non-elementary model checking problem. Work on identification of more efficient variants of quantified strategy logic includes [38], who formulate a variant with a 2-EXPTIME-complete model checking problem. In both cases, strategies are perfect recall strategies, rather than the imperfect recall strategies that form the basis for our PSPACE-completeness result for model checking.

Most closely related to this paper are a number of independently developed works that consider epistemic extensions of variants of strategy logic. Belardini [2] develops a logic, based on linear time temporal logic with epistemic operators, that adds an operator $\exists x_i$, the semantics of which existentially modifies the strategy associated to agent i in the current strategy profile. It omits the binding operator from [38], so provides no other way to refer to the variable x . The logic is shown to have nonelementary model checking complexity. This complexity is higher than the results we have presented because the semantics for strategies allows agents to have perfect information and perfect recall (though the semantics for the knowledge operators is based on imperfect information and no recall), whereas we have assumed imperfect information and no recall for strategies.

Another extension of strategy logic with epistemic operators has been independently developed by Čermák et al [9, 8]. Their syntax and semantics differs from ours in a number of respects. Although the syntax appears superficially in the form of an extension of LTL, it is more like CTL in some regards. The transition relation is deterministic in the sense that for each joint action, each state has a unique successor when that action is performed. Strategies are also assumed to be deterministic (whereas we allow nondeterministic strategies.) This means that, like CTL, the semantics of a for-

mula depends only on the current global state and the current strategy profile, whereas for LTL it is generally the case that the future structure of the run from a given global state can vary, and the truth value of the formula depends on how it does so. Although it seems that non-determinism could be modelled, as is commonly done, through the choice of actions of the environment, treated as an agent, the fact that strategies are deterministic, uniform and memoryless means that the environment must choose the same alternative each time a global state occurs in a run. This means that this standard approach to modelling of non-determinism does not work for this logic. The syntax of the logic moreover prevents epistemic operators from being applied to formulas with free strategy variables, whereas we allow fully recursive mixing of the constructs of our logic. Consequently, epistemic notions from our logic like $D_{\{i, \sigma(i)\}}$, expressing an agent's knowledge about the effects of its own strategy, which are used in several of our applications, do not appear to be expressible in this logic. Finally, the notion of "interpreted system" in this work, which corresponds most closely to our notion of "environment", also seems less general than our notion of environment because it defines the accessibility relations for the knowledge operators in a way that makes the environment state known to all agents.

In another paper [28], we have implemented a symbolic algorithm that handles model checking for the fragment $\text{CTLK}(Ags \cup \sigma(Ags))$, which, as shown above, encompasses the expressiveness of ATEL. Existing algorithms described in the literature for ATEL model checking [36, 7, 6] are based either on explicit-state model checking or are only partially symbolic in that they iterate over all strategies, explicitly represented. Our experimental results in [28] show that by comparison with the partially-symbolic approach, a fully-symbolic algorithm can greatly improve the performance and therefore scalability of model checking. The approach to model checking epistemic strategy logic implemented in [9, 8] is fully symbolic, but as already mentioned, this logic has a more limited expressive power than ours and its semantics does not permit representation of a nondeterministic environment. (It does not seem that the semantics could be extended to allow nondeterminism while retaining correctness of their algorithm.)

Our focus on this paper has been on an observational, or imperfect recall, semantics for knowledge. Other semantics for knowledge are also worth consideration, but are left for future work. We note one issue in relation to the connection to ATEL that we have established, should we consider a perfect recall version of our logic. ATEL operators effectively allow reference to situations in which agents switch their strategy after some actions have already been taken, whereas in our model an agent's strategy is fixed for the entire run. When switching to a new strategy, there is the possibility that the given state is not reachable under this new strategy. We have handled this issue in our translation by assuming that all states are initial, so that the run can be reinitialized if necessary to make the desired state reachable. This is consistent with an imperfect recall interpretation of ATEL, but it is not clear that this approach is available on a perfect recall interpretation. We leave a resolution of this issue to future work.

References

- [1] Rajeev Alur, Thomas A. Henzinger & Orna Kupferman (2002): *Alternating-Time Temporal Logic*. *Journal of the ACM* 49(5), pp. 672–713.
- [2] Francesco Belardinelli (2014): *Reasoning about Knowledge and Strategies: Epistemic Strategy Logic*. In: *Proc. 2nd Int. Workshop on Strategic Reasoning, SR 2014, Grenoble, France, April 5-6, 2014.*, pp. 27–33. ArXiv:1404.0837v1.
- [3] Patrick Blackburn & Jerry Seligman (1998): *What are hybrid languages?* In M. de Rijke, H. Wansing & M. Zakharyashev, editors: *Advances in Modal Logic*, 1, CSLI Publications, pp. 41–62.
- [4] Laura Bozzelli & Ruggero Lanotte (2010): *Complexity and succinctness issues for linear-time hybrid logics*. *Theoretical Computer Science* 411(2), pp. 454–469, doi:10.1016/j.tcs.2009.08.009.
- [5] Ronen I. Brafman, Jean-Claude Latombe, Yoram Moses & Yoav Shoham (1997): *Applications of a Logic of Knowledge to Motion Planning under Uncertainty*. *JACM* 44(5), pp. 633–668.
- [6] Simon Busard, Charles Pecheur, Hongyang Qu & Franco Raimondi (2013): *Reasoning about Strategies under Partial Observability and Fairness Constraints*. In: *1st International Workshop on Strategic Reasoning (SR2013)*, pp. 71–79.
- [7] Jan Caila, Dmitry Shkatov & Bernd-Holger Schlingloff (2010): *Finding Uniform Strategies for Multi-agent Systems*. In: *Computational Logic in Multi-Agent Systems (CLIMA XI)*, pp. 135–152.
- [8] Petr Čermák (2014): *A Model Checker for Strategy Logic*. MEng Individual Project thesis, Department of Computing, Imperial College London.
- [9] Petr Čermák, Alessio Lomuscio, Fabio Mogavero & Aniello Murano (2014): *MCMAS-SLK: A Model Checker for the Verification of Strategy Logic Specifications*. In: *26th International Conference, CAV 2014*, pp. 525–532.
- [10] Ashok K. Chandra, Dexter Kozen & Larry J. Stockmeyer (1981): *Alternation*. *Journal of the ACM* 28(1), pp. 114–133.
- [11] Krishnendu Chatterjee, Thomas A. Henzinger & Nir Piterman (2010): *Strategy logic*. *Information and Computation* 208(6), pp. 677–693, doi:10.1016/j.ic.2009.07.004.
- [12] Stephen Chong & Andrew C. Myers (2005): *Language-Based Information Erasure*. In: *IEEE Computer Security Foundations Workshop*, pp. 241–254.
- [13] Edmund M. Clarke, E. Allen Emerson & A. Prasad Sistla (1986): *Automatic verification of finite-state concurrent systems using temporal logic specifications*. *ACM Transactions on Programming Languages and Systems* 8(2), pp. 244–263.

- [14] Jan van Eijck (2013): *PDL as a Multi-Agent Strategy Logic*. In: *Proc. Conf. on Theoretical Aspects of Reasoning about Knowledge*. Published in CoRR, <http://arxiv.org/abs/1310.6437>.
- [15] E. Allen Emerson & Joseph Y. Halpern (1986): “*Sometimes*” and “*not never*” revisited: on branching versus linear time temporal logic. *Journal of the ACM* 33(1), pp. 151–178.
- [16] Kai Engelhardt, Peter Gammie & Ron van der Meyden (2007): *Model Checking Knowledge and Linear Time: PSPACE Cases*. In: *Proc. Symposium on Logical Foundations of Computer Science*, pp. 195–211.
- [17] Ronald Fagin, Joseph Y. Halpern, Yoram Moses & Moshe Y. Vardi (1995): *Reasoning About Knowledge*. The MIT Press.
- [18] Ronald Fagin, Joseph Y. Halpern, Yoram Moses & Moshe Y. Vardi (1997): *Knowledge-Based Programs*. *Distributed Computing* 10(4), pp. 199–225.
- [19] Massimo Franceschet & Maarten de Rijke (2006): *Model checking hybrid logics (with an application to semistructured data)*. *Journal of Applied Logic* 4(3), pp. 279–304, doi:10.1016/j.jal.2005.06.010.
- [20] Massimo Franceschet, Maarten de Rijke & Bernd-Holger Schlingloff (2003): *Hybrid Logics on Linear Structures: Expressivity and Complexity*. In: *10th International Symposium on Temporal Representation and Reasoning / 4th International Conference on Temporal Logic (TIME-ICTL 2003)*, pp. 166–173. Available at <http://doi.ieeecomputersociety.org/10.1109/TIME.2003.1214893>.
- [21] Joseph Y. Halpern & Yoram Moses (1990): *Knowledge and Common Knowledge in a Distributed Environment*. *Journal of the ACM* 37(3), pp. 549–587.
- [22] Joseph Y. Halpern & Yoram Moses (2007): *Characterizing Solution Concepts in Games Using Knowledge-Based Programs*. In: *the 20nd International Joint Conference on Artificial Intelligence (IJCAI2007)*, pp. 1300–1307.
- [23] Joseph Y. Halpern & Kevin R. O’Neill (2008): *Secrecy in Multiagent Systems*. *ACM Transactions on Information and System Security* 12(1).
- [24] Joseph Y. Halpern & Nan Rong (2010): *Cooperative Equilibrium (Extended Abstract)*. In: *9th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS’10)*, pp. 1465–1466.
- [25] Jens Ulrik Hansen (2011): *A Hybrid Public Announcement Logic with Distributed Knowledge*. *Electronic Notes in Theoretical Computer Science* 273, pp. 33–50, doi:10.1016/j.entcs.2011.06.011.
- [26] Wiebe van der Hoek & Michael Wooldridge (2002): *Tractable multiagent planning for epistemic goals*. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS’02)*, pp. 1167–1174.

- [27] John F. Horty. (2001): *Agency and deontic logic*. Oxford University Press.
- [28] Xiaowei Huang & Ron van der Meyden (2014): *Symbolic Model Checking Epistemic Strategy Logic*. In: *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, pp. 1426–1432.
- [29] Xiaowei Huang & Ron van der Meyden (2014): *A Temporal Logic of Strategic Knowledge*. In: *14th International Conference on Principles of Knowledge Representation and Reasoning (KR2014)*.
- [30] Wojciech Jamroga (2003): *Some Remarks on Alternating Temporal Epistemic Logic*. In: *Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS 2003)*.
- [31] Wojciech Jamroga & Thomas Ågotnes (2007): *Constructive knowledge: what agents can achieve under imperfect information*. *Journal of Applied Non-Classical Logics* 17(4), pp. 423–475, doi:10.3166/jancl.17.423-475.
- [32] Wojciech Jamroga, Thomas Ågotnes & Wiebe van der Hoek (2008): *A Simpler Semantics for Abilities under Uncertainty*. Technical Report, Clausthal University of Technology.
- [33] Wojciech Jamroga & Jürgen Dix (2006): *Model Checking Abilities under Incomplete Information Is Indeed Delta2-complete*. In: *the 4th European Workshop on Multi-Agent Systems (EUMAS’06)*.
- [34] Wojciech Jamroga & Wiebe van der Hoek (2004): *Agents that Know How to Play*. *Fundamenta Informaticae* 62, pp. 1–35.
- [35] Geert Jonker (2003): *Feasible strategies in alternating-time temporal*. Master’s thesis, University of Utrecht, The Netherlands.
- [36] Alessio Lomuscio & Franco Raimondi (2006): *Model Checking Knowledge, Strategies, and Games in Multi-Agent Systems*. In: *the proceedings of the 5th international joint conference on Autonomous agents and multiagent systems (AAMAS 2006)*, pp. 161–168.
- [37] Ron van der Meyden (1996): *Knowledge Based Programs: On the Complexity of Perfect Recall in Finite Environments*. In: *6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1996)*, pp. 31–49.
- [38] Fabio Mogavero, Aniello Murano & Moshe Y. Vardi (2010): *Reasoning About Strategies*. In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, pp. 133–144, doi:10.4230/LIPIcs.FSTTCS.2010.133.
- [39] Sieuwert van Otterloo & Geert Jonker (2005): *On Epistemic Temporal Strategic Logic*. *Electronic Notes in Theoretical Computer Science* 126, pp. 77–92, doi:10.1016/j.entcs.2004.11.014.

- [40] Rohit Parikh (1983): *Propositional Game Logic*. In: *IEEE Symp. on Foundations of Computer Science*, pp. 195–200.
- [41] Rohit Parikh & Ramaswamy Ramanujam (1985): *Distributed Processes and the Logic of Knowledge*. In: *Logics of Programs 1985*, pp. 256–268.
- [42] Marc Pauly (2002): *A modal logic for coalitional power in games*. *Journal of Logic and Computation* 12(1), pp. 149–166.
- [43] Amir Pnueli (1977): *The Temporal Logic of Programs*. In: *Symp. on Foundations of Computer Science*, pp. 46–57.
- [44] Ramaswamy Ramanujam & Sunil Easaw Simon (2008): *Dynamic Logic on Games with Structured Strategies*. In: *Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR2008)*, pp. 49–58.
- [45] Olivier Roy (2009): *A dynamic-epistemic hybrid logic for intentions and information changes in strategic games*. *Synthese* 171(2), pp. 291–320, doi:10.1007/s11229-009-9644-3.
- [46] Henning Schnoor (2010): *Explicit Strategies and Quantification for ATL with Incomplete Information and Probabilistic Games*. Technical Report 1008, Institut für Informatik, Christian-Albrechts Universität zu Kiel.
- [47] Pierre-Yves Schobbens (2004): *Alternating-time logic with imperfect recall*. *Electronic Notes in Theoretical Computer Science* 85(2), pp. 82–93, doi:10.1016/S1571-0661(05)82604-0.
- [48] Thomas Schwentick & Volker Weber (2007): *Bounded-Variable Fragments of Hybrid Logics*. In: *Proc. STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science*, Springer LNCS 4393, pp. 561–572, doi:10.1007/978-3-540-70918-3_48.
- [49] A. Prasad Sistla & Edmund M. Clarke (1985): *The Complexity of Propositional Linear Temporal Logics*. *Journal of the ACM* 32(3), pp. 733–749.
- [50] Moshe Y. Vardi (1996): *Implementing Knowledge-Based Programs*. In: *the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, pp. 15–30.
- [51] Wiebe van der Hoek, Wojciech Jamroga & Michael Wooldridge (2005): *A logic for strategic reasoning*. In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS’05)*, pp. 157–164.
- [52] J. Todd Wittbold & Dale M. Johnson (1990): *Information flow in nondeterministic systems*. In: *Proc. IEEE Symp. on Security and Privacy*, pp. 144–161.
- [53] Steve Zdancewic & Andrew C. Myers (2001): *Robust Declassification*. In: *IEEE Computer Security Foundations Workshop*, p. 15.